Airport Navigation Aid Database Application 2.0 (AIRNAV 2.0)

AirNav Web Services Use Cases and Business Rules

Revision History

#	Version	Date	Description	By
1	V00R01	12/04/2007	Draft Version of the Document	Frances K. Hubbard

Table of Content

1 Introduction

This document fully describes the functionality of Maintain Web Services within the AIRNAV 2.0 system. These requirements are captured in the AVN iSM Use Case format. It details the needs the system must address to capture and provide data related to AIRNAV 2.0 system.

Refer to AIRNAV interface control document for implementation requirements related to Web Services.

1.1 Abbreviations and Acronyms

Refer the document AIRNAV - Glossary for abbreviations, acronyms and other general terminology used in the AIRNAV documentation.

2 Use Cases

The Maintain Web Services module will include following use cases:

1. Search Aeronautical Data
2. Add Aeronautical Data
3. Edit Aeronautical Data
4. Delete Aeronautical Data

The details of each of the above mentioned use cases are described in this document.

2.1 Use Case Specification: Search Aeronautical Data

2.1.1 Brief Description

This use case describes the process of searching for aeronautical data record(s) in AIRNAV system. See AIRNAV Interface Control Documents for implementation details.

2.1.2 Actors

Following are the actors for this use case:

1. Web Services User

2.1.3 Pre – conditions

1. User must have access privileges to AVN Web Services.

2.1.4 Basic Flow of Events

1. System receives a request for aeronautical data.
2. System interprets XML based request
3. System queries database for response.
4. System returns response in an XML format.
5. If no records satisfy the request, a standard response xml will be composed.
6. System returns response in an XML format.

2.1.5 Alternate Flows

There are no alternate flows in this use case.

2.1.6 Sub – flows

There are no sub-flows for this use case.

2.1.7 Key Scenarios

There are no key scenarios for this use case.

2.1.8 Post – conditions

There are no post conditions for this use case.

2.1.9 Extension Points

There are no extension points for this use case.

2.1.10 Special Requirements

The requesting web service should have the ability to indicate if the response should have a Summary or Detail level of data.

2.1.11 Additional Information

There is no additional information for this use case.

2.1.12 Business Rules

This function will be used for all requests for aeronautical data.

2.2 Use Case Specification: Add Aeronautical Data

2.2.1 Brief Description

This use case describes the process of adding aeronautical data via a Web Service. See AIRNAV Interface Control Documents for implementation details.

2.2.2 Actors

There is only one actor for this use case:

- Web Services User

2.2.3 Pre – conditions

User must have access privileges to AVN Web Services.

2.2.4 Basic Flow of Events

1. System receives a request for addition of aeronautical data.
2. System interprets XML based request
3. System validates and adds data to database.
4. If system is unable to add data to database, a standard response xml will be composed.
5. System returns response in an XML format.

2.2.5 Alternate Flows

There are no alternate flows in this use case.

2.2.6 Sub – flows

There are no sub-flows for this use case.

2.2.7 Key Scenarios

There are no key scenarios for this use case.

2.2.8 Post – conditions

1. A new aeronautical record is added to the database and is searchable.

2.2.9 Extension Points

There are no extension points for this use case.

2.2.10 Special Requirements

There are no special requirements for this use case.

2.2.11 Additional Information

There is no additional information for this use case.

2.2.12 Business Rules

This function will be used for all requests to add aeronautical data to the database.

2.3 Use Case Specification: Edit Aeronautical Data

2.3.1 Brief Description

This use case describes the process of editing an existing aeronautical via a Web Service. See AIRNAV Interface Control Documents for implementation details.

2.3.2 Actors

Following are the actors for this use case:

- Web Services User

2.3.3 Pre – conditions

User must have access privileges to AVN Web Services.

2.3.4 Basic Flow of Events

1. System receives a request for update of aeronautical data.
1. System interprets XML based request.
2. System invokes a service to the IFPA Enterprise requesting associated record(s) to the requested record.
3. Web Service returns no associated record(s).
4. System validates and edits aeronautical record.
5. If system is unable to edit data, a standard response xml will be composed.
6. System returns response in an XML format.

2.3.5 Alternate Flows

1. System receives a request for update of aeronautical data.
2. System interprets XML based request.
3. System invokes a service to the IFPA Enterprise requesting associated record(s) to the requested record.
4. IFPA Enterprise returns a list of associated record(s) and their respective owner(s) that are linked to the record.
5. System does not allow the update of the aeronautical record
6. System composes a negative response and returns the list of associated record(s) and their respective owner(s)
7. System returns response in an XML format.

2.3.6 Sub – flows

There are no sub-flows for this use case.

2.3.7 Key Scenarios

There are no key scenarios for this use case.

2.3.8 Post – condition

An aeronautical record is edited in the database and is searchable.

2.3.9 Extension Points

There are no extension points for this use case.

2.3.10 Special Requirements

The requesting web service should have the ability to indicate if a returning list of associated records is needed.

2.3.11 Additional Information

There is no additional information for this use case.

2.3.12 Business Rules

This function will be used for all requests to edit aeronautical data.

2.4 Use Case Specification: Delete Aeronautical Data

2.4.1 Brief Description

This use case describes the process of deleting an existing aeronautical data record. The delete refers to the physical deletion of the record.

2.4.2 Actors

There is only one actor for this use case:

- Web Services User

2.4.3 Pre – conditions

User must have access privileges to AVN Web Services.

2.4.4 Basic Flow of Events

1. System receives a request for deletion of aeronautical data.
2. System interprets XML based request.
3. System invokes a service to the IFPA Enterprise requesting associated record(s) to the requested record.
4. Web Service returns no associated record(s).
5. System validates and deletes aeronautical record.
6. If system is unable to delete data, a standard response xml will be composed.
7. System returns response in an XML format.

2.4.5 Alternate Flows

1. System receives a request for deletion of an aeronautical record.
2. System interprets XML based request.
3. System invokes a service to the IFPA Enterprise requesting associated record(s) to the requested record.
4. IFPA Enterprise returns a list of associated record(s) and their respective owner(s) that are linked to the record.
5. System does not allow the deletion of the record and composes a list of associated record(s) and their respective owner(s).
6. System returns a negative response and the list of associated records in an XML format.

2.4.6 Sub – flows

There are no sub-flows for this use case.

2.4.7 Key Scenarios

There are no key scenarios for this use case.

2.4.8 Post – conditions

1. The selected aeronautical record will be deleted from the system.
2. The deleted aeronautical record will not be searchable in the system.

2.4.9 Extension Points

There are no extension points for this use case.

2.4.10 Special Requirements

The requesting web service should have the ability to indicate if a returning list associated records is needed.

2.4.11 Additional Information

There is no additional information for this use case.

2.4.12 Business Rules

This function will be used for all requests to delete aeronautical data.

AIRNAV Web Services

1.0 Airport / Runway / Lighting Services

1.1 Airport / Runway / Lighting Query Summary

Needed for application pre-selection

1.2 Airport / Runway / Lighting Query Detail

Full information return

1.3 Airport / Runway / Lighting Add

Needed for applications to add aeronautical data into AIRNAV

1.4 Airport / Runway / Lighting Update

Needed for applications to edit aeronautical data in AIRNAV

1.5 Airport / Runway / Lighting Delete

Needed for applications to delete aeronautical data in AIRNAV

2.0 Navaid and Component Services

2.1 Navaid & Component Query Summary

Needed for application pre-selection.1

2.2 Navaid & Component Query Detail

Full information return

2.3 Navaid & Component Add

Needed for applications to add aeronautical data into AIRNAV

2.4 Navaid & Component Update

Needed for applications to edit aeronautical data into AIRNAV

2.5 Navaid & Component Delete

Needed for applications to delete aeronautical data in AIRNAV

3.0 Expanded Service Volume Services

ESV services will have to be implemented in both AIRNAV and the ESVMS system.

3.1 ESV Query

Needed to query the ESVMS system

3.2 ESV Add

Needed to add ESV request in the ESVMS system

3.3 ESV Update

Needed to edit an ESV from IFPA

3.4 ESV Delete

Needed to delete an ESV from IFPA

4.0 Obstacle Services

Obstacle data will be maintained in AIRNAV until the Obstacle Repository System (ORS) is implemented with flight procedure development capability.

4.1 Obstacle Query

Full information return

4.2 Obstacle Add

Needed for applications to add aeronautical data into AIRNAV

4.3 Obstacle Update

Needed for applications to edit aeronautical data into AIRNAV

4.4 Obstacle Delete

Needed for applications to delete aeronautical data from AIRNAV

5.0 Information Services

5.1 SIAP Query

Execute request for SIAP procedures that use Airport / Runway / or Navaid records. This service is needed for data products and for determining if an AIRNAV record can be updated or deleted.

5.2 FIX Query

Execute request for FIX's that use Airport / Runway / or Navaid records. This service is needed for data products and for determining if an AIRNAV record can be updated or deleted.

5.3 MSA Query

Execute request for FIX's that use Airport / Runway / or Navaid records. This service is needed for data products and for determining if an AIRNAV record can be updated or deleted.

5. Navaid Downgrade Trigger

Execute request for affected Active procedures, fixes, etc when a navaid system or component is cancelled from a system.

Airport Navigation Aid Database Application 2.0 (AIRNAV 2.0)

AVN Web Services

Interface Control Document

Revision History

#	Version	Date	Description	By
1	V00R01	12/04/2007	Draft Version of the Document	Frances K. Hubbard

Table of Content

1 Introduction

This document fully describes the functionality of Maintain Web Services within the AIRNAV 2.0 system. These requirements are captured in the AVN ISM Use Case format. It details the needs the system must address to capture and provide data related to AIRNAV 2.0 system.

1.1 Abbreviations and Acronyms

Refer to the document AIRNAV - Glossary for abbreviations, acronyms and other general terminology used in the AIRNAV documentation.

1.2 Identification

This is the Interface Control Document (ICD) for the AVN Web Services (AVN-WS). It is a technical software interface document that specifies a set of web services, and the usage of these web services

An AVN-WS interface version package comprises of this ICD, the Web Services Definition Language (WSDL) file and the XML Schema Definition (XSD) files that implement the interface. Information and downloads can be found at the following web site:

http://www.aixm.aero/public/subsite_homepage/homepage.html

The purpose of this document is to:

- Provide a complete definition of the web services interface and the XML schema files used for interaction between any client and the FAA Enterprise.
- Explain how the web services are intended to be used in practice, on both sides of the interface.
- Provide a definition of the web services that will be used for interaction between the AirNav Application and an outside data consumer / provider.

1.3 Intended Audience

This document is written for a technical audience, and assumes familiarity with web services related concepts and standards:

- **FAA AVN Information Technology Team** – use this document to design and implement the required web services in the FAA AVN Enterprise
- **Contractor AirNav Systems and Software Engineers** – use this document to design and implement the AirNav Application interfaces.

1.4 Document Scope

This document describes all the web services that are used by the AirNav Application. It defines:

- ✓ Web service operations in terms of their request, response and fault messages
- ✓ Web services standards used for the transport and encoding of messages

1.5 Interface Version Package

An AVN-WS Interface Version package is defined as a zipped package, named AVN_WebService_M_N.zip, of the set of artifacts listed in Table 1.5, each of which are themselves individually versioned. The version control over the interface schema files is maintained by the FAA.

Table 1.5 Interface Version Package

File	**Description**
AVN Web Services Interface Control Document	This document.
DataAccessService_M_N.wsdl	WSDL file. The top-level code description of the interface that identifies the web service Operations and the Messages they support
DataAccessServiceMessages_M_N.xsd	XML Schema file. Defines the XML Schema for the request, response and fault messages for each of the web service Operations. This file is referenced from the WSDL.
AIXM-5 Payload Data	
	XML Schema file. Defines the object/data structures for the AIXM-5 data to be transferred. These files are referenced from the Message schema file. Higher-level .xsd files may include lower-level .xsd files that define third party base types (e.g., GML .xsd files) and common types.
TBD: FAA_extension_xxx.xsd	FAA Extensions to AIXM-5
AIXM-Feature.xsd	Core aeronautical data model supporting international air navigation
AIXM-DataTypes.xsd	Common set of data types and values domains used for aeronautical data
AIXM-AbstractGML-ObjectTypes.xsd	Base type for AIXM object and feature types
profile/gml4aixm.xsd	GML Subset schema for AIXM
xlink/xlinks.xsd	GML 3.0 candidate xlinks schema
Object Payload Data	
TBD	XML Schema file. Defines the object/data structures for the object data to be transferred. These files are referenced from the Message schema file. There may be different schema files for different object types (e.g., ESV, Agreements, etc).

2 Documents

2.1 Governing Source Documents

The following documents are referenced as governing source documents within this particular document. Governing Source documents serve to establish the interface baseline.

Table 2.1.1 Governing Source Documents

Reference #	Document #	Organization	Title
AIXM-5	*AIXM-5 Release Candidate 1(RC1) TBD: AIXM-5 Release Candidate*	Eurocontrol	UML and XSD Schemas Files
	2(RC2) will be released in mid-2007. The ICD will be baselined to RC2.		
AIXM-5-FAA	See Section 1.6	FAA	XSD Schemas Files
FAA-SDS	None provided.	FAA ATO AVN-313A Data Management Team	Sensitive Data Security Administration for AVN, Version 1.2 (document markings indicate Version 1.1, December 14, 2005).
SOAP	http://www.w3.org/TR/soap12-part1/	World Wide Web Consortium (W3C)	SOAP Version 1.2, W3C Recommendation 24 June 2003.
WS-I	http://www.ws-i.org/Profiles/BasicProfile-1.02004-04-16.html	Web Services Interoperability Organization	WS-I Basic Profile Version 1.0, Final Material, 2004-04-16
WSDL	http://www.w3.org/TR/2001/NOTE-wsdl-20010315	World Wide Web Consortium (W3C)	Web Services Description Language 1.1, W3C Note 15 March 2001.
XSD	http://www.w3.org/TR/xmlschema-0/	World Wide Web Consortium (W3C)	XML Schema 1.0, W3C Recommendation 28 October 2004.

2.2 Reference Documents

Reference documents provide background and/or supplementary information to the contents of this document.

Table 2.2.1 Reference Documents

Reference	Organization	Title	Notes
AIXM-5-MGRD	Eurocontrol	AIXM 5 Exchange Model Goals, Requirements and Design. 2006/06/22	Defines objectives, framework and key design decisions for AIXM 5.
AIXM-5-CONV	Eurocontrol	Data Modeling Conventions for AIXM 5. 2006/06/22	Defines UML conventions.
GML / ISO/TC 211/WG 4/PT 19136 OGC GML RWG	ISO	Geography Markup Language (GML). Version 3.1.0. 2004-Feb-7.	GML referenced by AIXM 5 specification. Note: AIXM-5 references the GML 3.1.1 schema.
WFS / OGC 04-094	Open Geospatial Consortium Inc	Web Feature Service Implementation Specification. Version: 1.1.0. Date: 2005 May-3	Defines interfaces for data access and manipulation operations on geographic features using HTTP as the distributed computing platform. Via these interfaces, a web user or service can combine, use and manage geodata.
WFS / OGC 06-027r1	Open Geospatial Consortium Inc.	Corrigendum for the OpenGIS Web Feature Service (WFS) Implementation Specification.	
Filter / OGC 04-095	Open Geospatial Consortium Inc	OpenGIS Filter Encoding Implementation Specification. Version: 1.1.0. Date: 2005 May-3	Describes an XML encoding of the OGC Common Catalog Query Language (CQL) as a system neutral representation of a query predicate.

3 Overview

This section establishes the context for understanding the interface design by describing the web service model, operations and standards upon which the solution is based.

3.1 Design Objectives

1. **Support the IFPA Application Operator's data access needs**
   Data access needs of IFPA application (SIAP, FIX, IPDS, etc) are extensive and must be met by Airnav 2.0. A flight procedure work area could include up to 200 nautical miles of aeronautical data.

This includes the following activities against a Data Source (Data Server):

- Discovery: browsing, filtering, and searching data;
- Access: retrieving data to populate a flight procedure development work area, refreshing a work area, or submitting new or changed data from a work area back to the Data Source.

2. **Achieve interactive response times for IFPA Application Operator's discovery activities**

   Response time is a function of

   - Network infrastructure capacity: network latency, network bandwidth;
   - Data Source server capacity and performance;
   - Application client software performance and Data Source software performance;
   - Nature of the web services interface design.

   The first three factors are outside the scope of this document, but clearly have substantial bearing on the performance and what the Operator experiences. The web services interface is designed to be lightweight and also to allow service performance to be tuned to some extent by providing request parameters that control the amount of server effort required (whether queries need to be performed for associated objects) and the level of detail (data volume) returned.

3. **Give the Application Operator visibility and control over large data volumes**

   For large data volumes, the Operator may need to wait for data to be transferred between a Data Source and the Application. This web service interface design permits selective data transfer through the following mechanisms:

   - GetGmlObject operation to enable retrieval of specific objects
   - Separate Get/Post from/to URL for retrieval and submission of file attachments (document, screen capture image, etc.)
   - Filter parameters on Get operations to reduce the number of objects found and returned.
   - Control over the level of detail returned for an object in a Get operation,

4. **Achieve graceful degradation of service under heavy service loads**

   If a server is inundated with requests that exceed its resources to fulfill, service must not be completely lost, rather requests should be fulfilled, albeit delayed. The interface needs to be designed to degrade gracefully when loads are heavy.

5. **Leverage existing standards, technology and tools appropriately**

   Existing open standards are used wherever possible to avoid "re-inventing the wheel". Standards that are well supported by tools, prevalent in use, and have good expected longevity are preferred.

6. **Establish well-defined, controlled interfaces**

   The design accomplishes this through

• Defining a web service interface version, with version controlled artifacts: WSDLs, XSDs.
• This ICD, which documents the content and intent of a particular web service interface.

7. **Minimize interface software changes due to changes in the transfer data Model**
General purpose operations should be designed so that new data (object types or configuration data types) can be introduced without having to add new web service operations. This reduces the effort required to change the interface and the supporting software.

3.2 Web Services Operations

Web services connect distributed software applications over a network, typically by passing data in eXtended Markup Language (XML) between clients and servers using the HyperText Transport Protocol (HTTP). IFPA Application web services conform to the Simple Object Access Protocol (SOAP), which defines a framework for XML messages in the web service request and response.

Table 3-2 provides an overview of each of the web service operations in terms of what it does, and the technologies used to implement it.

The GetFeature and GetGmlObject services are implementations of the Open Geospatial Consortium (OGC) Web Feature Service Implementation Specification [WFS] standard. This ICD tailors and constrains the [WFS] standard to support the required service functionality.

Table 3.2 Web Service Operations

Operation	Description	Implementation Technologies
1. GetGMLObject	Used to retrieve/refresh a set of aeronautical features from a Data Server using object identifiers. Intended for retrieving a specific aeronautical feature version or a set of versions of a feature.	SOAP, conforming to WS-I Basic Profile, over HTTP. Based on the OGC WFS Specification [WFS]. Aeronautical features encoded in AIXM 5.
2. GetFeature	Used to retrieve/refresh a set of aeronautical features from a Data Server using a spatial, temporal, and textual filter parameters. Intended for retrieving aeronautical features in a geographical area – single or multiple versions.	SOAP, conforming to WS-I Basic Profile, over HTTP. Based on the OGC WFS Specification [WFS] and OGC Filter Encoding Implementation Specification [FILTER] Aeronautical features encoded in AIXM 5.
GetObject	Used to retrieve/refresh a set of objects from a Data Server using a spatial, temporal, and textual filter parameters. Intended for retrieving non-aeronautical objects in a geographical area or through other filter constraints – single or multiple versions.	SOAP, conforming to WS-I Basic Profile, over HTTP. Based on the OGC Filter Encoding Implementation Specification [FILTER]
PutFeature	Used to store aeronautical features back to a Data Server. Submits one 'Transactable Object Set', a tightly coupled set of objects that are stored in one database transaction.	SOAP, conforming to WS-I Basic Profile, over HTTP. Aeronautical objects encoded in AIXM 5.
PutObject	Used to store objects back to a Data Server. Intended for submitting non-aeronautical objects. Submits one 'Transactable Object Set', a tightly coupled set of objects that are stored in one database transaction.	SOAP, conforming to WS-I Basic Profile, over HTTP.
GetAttachment	Used to retrieve an attachment to an object from a Data Server. An attachment can be any type of file, that has been associated to an object.	Get from URL over HTTP.
PostAttachment	Used to store an attachment to an object to a Data Server. An attachment can be any type of file.	Post to URL over HTTP.
GetConfigData	Used to provide new or updated configuration data.	SOAP, conforming to WS-I Basic Profile, over HTTP.

3.3 Web Services Definition Model

All the Web Services are SOAP-based services follow the Web Services Definition Language (WSDL) model. A WSDL artifact describes a web service as a set of one or more SOAP-based Operations.

3.4 Soap Model

The SOAP message is embedded within an HTTP request or response. The SOAP Message consists of an Envelope element and optionally, one or more attachments that follow. The Envelope element has Header and Body elements. The SOAP Header serves to guide the routing of SOAP messages, and is not of concern to this interface. The SOAP Body contains the application payload, i.e., the input (request), output (response) or fault message as declared in the WSDL for the operation.

3.5 Attachments

Attachments are files (of any type) that are associated to an Object. For example, an airport may have many file attachments, such as surveys, letters, etc. As attachments may be large files, they are handled using a handshake mechanism between the client application and the FAA Enterprise. The two-stage process involves an initial transfer of metadata describing the attachment to transfer, followed by a get from or a put to, a URL specified by the Enterprise.

The necessary metadata includes the following attributes: mime-type, file name, file creator, file size and last modified date. Optionally the metadata contains the referral URL to get or put the data to or from. This URL need only be provided by the Enterprise and not by the client application. A UID is provided when attachment information is received from the Enterprise. For obvious reasons, this is not present when new attachments are being created.

The handshake mechanism is orchestrated by the client application storing the data within the Enterprise. The process assumes a decoupling of file storage to an external file server and not within the database application itself. The UID of the file is stored in the database application itself, and this UID are used to coordinate with the file storage system for retrievals, updates and deletions. The orchestration is discussed in the three following use cases: Retrieve, Update and Create attachment.

TBD: The proposed file storage component for attachments in the Enterprise is Oracle Files, a component of Oracle Collaboration Suite.

3.5.1 Retrieve Attachment

1. Client application requests data using GetFeature or GetGmlObject web service from FAA Enterprise.

2. For any attachments associated with the returned feature collection, the Enterprise application coordinates using a UID for the attachment with the file storage service for all metadata regarding that file as well as the referral URL to retrieve the file from.

3. The GetFeature or GetGmlObject response message indicates attachments exist for retrieved data complete with necessary metadata. The referral URL is present in metadata. (For AIXM-5 features, the attachment metadata is implemented as a Note object).

4. Client application retrieves attachments using HTTP GET from the referral URL and uses the metadata to store locally.

3.5.2 Update Attachment

1. Client application updates a feature to Enterprise using PutFeature web service.

2. The client application submits data to PutObject web service including Metadata for the updated attachment(s) associated with a feature. The PutObject request indicates if the attachment is to be updated (replace the existing attachment with the modified attachment) or amended (keep the existing attachment, and create a new version of the attachment).

3. Enterprise application receives message. For each attachment listed, coordinates with the file storage service using the UID for the attachment as well as all up to date metadata. File storage service responds with up to date metadata plus referral URL for all files that have changed and thus require a submit back to the enterprise with the update.

4. Response XML to client includes full metadata for each attachment to be updated only, including referral URLs for data to be put to.

5. Client application sends data through HTTP POST to the referral URL.

3.5.3 Create New Attachment

1. Client application stores a feature to Enterprise using PutFeature web service

2. The client application submits data to PutObject web service including metadata for the new attachment(s) associated with a feature.

3. Enterprise application receives message. For each attachment listed that is not yet present, coordinates with the file storage service by sending the metadata for the file. File storage service creates a placeholder record for the file and return the UID as well as all metadata for each file.

4. The PutObject response XML to client application includes full metadata for each attachment to be updated, including referral URLs for data to be put to.

5. Client Application sends data through HTTP POST to the referral URL.

3.6 Compression

Payload data is transmitted through the interface as an XML Schema *base64binary* blob. This blob may optionally be compressed by the data source. The data source must specify what type of data the blob contains using a *mime type* as well as if the data is being transmitted in a compressed format within the binary or not. The receiver of the data must then use the cues listed by the data source to decide if the binary data must be uncompressed before use or not.

3.7 Security

3.7.1 Internal Web Services

FAA ATO Data Security Policy [FAA-SDS 6] requires that all external web services interface into a security infrastructure and conform to WS-Security specifications.

4 Message Design

This section describes the key design concepts found in the interface and the general patterns followed by request, response and fault messages.

4.1 Web Service Interface Version

This web service interface is versioned to control the deployment and changes to the interface. To evolve and advance the interface, a new interface version must be created and deployed.

A client Application and the Data Source's web service server communicate using a specific version of the web service interface. When a new interface version is introduced, both the client and server platforms must be able to accommodate the change. For any significant change, this requires deploying new software releases on both the client and server.

If multiple releases of the client application need to be supported at one time, then the Data Source's service must support multiple interface versions. This situation is most likely to occur during transition periods where a new release of the client application is being rolled out.

4.1.1 Interface Version Levels

An Interface Version is identified by a **major** version number (M) and a **minor** version number (N), i.e., M.N, M/N, or _M_N.
If an interface change is significant, disruptive, or not-backwardly compatible, then it is a major revision to the interface, incrementing M. Most likely a major revision to the interface will coincide with a major release of the IFPA Applications.

If an interface change is backwardly compatible with deployed client and server software releases, then it should be considered a minor revision to the interface, e.g., Release 1.1, 1.2, etc. Changes can be made to the WSDL or XML Schema files already deployed in such a way as to not break the deployed client or server software. For example, in a situation where expanding the definition of a XML Schema enumeration to include another value does not affect any software.

The situation may occur where the Service Provider must support more than one minor interface revision and have different logic for each. To easily support this style of coding and to support software diagnosis, the Interface Version is included as an element of the request, response and fault header blocks

4.1.2 Interface Version Package

A Web Service Interface Version is defined as a zipped package, named AVN_WebService_M_N.zip, of the set of artifacts listed in Table 1-2, each of which are themselves individually versioned.

4.2 Objects

The payload data is conceptually organized into **Objects**. The key characteristics of an object are as follows:

- **Properties:** every object contains a set of properties (also referred to as attributes).
- **ObjectType:** identifies a group of objects that share the identical set of object properties, e.g., the Workflow ObjectTypes are Request, Project and Task.
- **ObjectID:** uniquely distinguishes the physical entity represented by the object.

The ObjectID is expected to be a value machine-generated by a Data Source (referenced by a **Namespace**) so that the object can be uniquely identified within the Data Source.

- **ObjectVersion:** as the properties of an object may vary over time or more than one definition may exist simultaneously, the ObjectVersion is used to define an instance of an object.
- **ObjectName:** as the ObjectID may be machine-generated and not humanly recognizable, the ObjectName is used to capture the natural key or some form of name.
- **Namespace:** uniquely identifies a Data Source, and possibly a context within The Data Source, within which it can be certain that the ObjectID is unique (in AIXM-5 this is referred to as the codespace).

The AIXM-5 model and its XML encoding is used to represent aeronautical and instrument procedure data. An object within AIXM-5 is referred to as a Feature, a Feature timeslice defines a specific instance of a Feature. The AIXM-5 object model does not cover all the data concepts that are required. Hence the XML schema for some ObjectTypes is defined independently of AIXM-5.

4.2.1 Object Reference

Object references are needed to implement associations between objects for the purpose of:

- Browsing across related objects through an Application, e.g., runways, instrument procedures and NAVAIDs related to an airport.

- Traversing object dependencies, e.g., so that all the object dependencies – Fixes, Navaids, Obstacles, Runway, Airport – can be found and retrieved.

An object reference requires the following information:

1. Namespace
2. ObjectType
3. ObjectID
4. ObjectVersion(s), if applicable
5. ObjectName

Object references are encoded as xlink statements. This is consistent with [WFS] and AIXM-5, which use the xlink standard for feature associations.

4.2.1.1 Object References to AIXM-5 Features

In AIXM-5 the ObjectID is represented by the aixm:identifier element, which uniquely identifies a feature with an identifier and codespace attributes. The ObjectVersion is represented by the version element, which is an FAA extension to all AIXM-5 feature timeslices.

If an Object has an ObjectID, represented by the aixm:identifier element, then references to it using an Xlinks follows these rules:

1. The xlink href attribute complies with XPath 1.0 and reference an object through its ObjectType, Namespace (aixm:identifier codespace attribute), ObjectId (aixm:identifier element), and ObjectVersion(s) (version element).

2. An xlink href attribute may reference one or more versions of an object. This allows for temporal associations. For example, runway version 4 may be associated to aerodrome versions 5, 6 and 7.

3. The xlink href attribute also includes an alternate 'natural key' (also an Xpath reference) to the object. This allows object references to be resolved across multiple data sources.

4. The xlink title attribute contains the ObjectName.

There will be cases where an Object has no ObjectID represented by the aixm:identifier attribute. For example, an RVR is a feature in AIXM-5, but is stored as part of a runway object in the FAA Enterprise, and so does not have its own ObjectID (control number in the FAA Enterprise). When an Object has no aixm:identifier, the features are associated within the message payload through an xlink statement, following these rules:

1. The xlink href attribute complies with XPath 1.0 and reference a feature instance in the XML document by its gml:id. (In AIXM-5, all features and feature instances have a gml:id, which is of XML type ID, so is constrained to be unique in the XML document within which it occurs).

For example, When a request is made for a runway object, the AVN WS will return an AIXM-5 runway feature and an RVR feature. The RVR feature references its associated Runway.

4.2.1.2 Object Reference

References to non-AIXM-5 features using an Xlink follows these rules:

1. The xlink href attribute complies with XPath 1.0 and reference an object through its Namespace, ObjectType, ObjectId (control number), and ObjectVersion(s). The attributes will be dependent on the object type.

2. The xlink href attribute includes an alternate 'natural key' if applicable (also an Xpath reference) to the object. This allows object references to be resolved across

multiple data sources.

3. The xlink title attribute contains the ObjectName.

4.2.2 Object Directory

To support client application browsing of the data objects held by a Data Source (using the GetObject operation), the Data Source must offer up some top-level groupings of objects. The **Object Directory** concept serves this purpose, aggregating a set of objects, much like a file folder or directory. An Object Directory may contain other Object Directories to form a hierarchy, much like a computer's file system. This follows a Representational State Transfer (REST) architecture model, wherein each response is linked to more data - which may be retrieved by traversing the link. Each response allows the client to drill down to get more detailed information.

Object Directories may only be retrieved, they cannot be stored via the PutObject operation, Object Directories can be requested from a Data Source by setting the ObjectType parameter to 'ObjectDirectory' in the GetObject operation

4.3 Object Filters, Detail and Associations

4.3.1 Filters

The GetFeature and GetObject operations enable data to be filtered on the server before it is sent back to the client. Each operation has its own specific filter parameters, e.g., spatial location, date, etc. Filters are implemented in the request messages using the OpenGIS Filter Encoding Implementation Specification [FILTER].

4.3.2 Detail

To ensure that client application browsing using the GetFeature, GetGmlObject and GetObject operations are responsive, server-side processing time and data transfer volumes need to be kept to a reasonable size. The amount of information returned is controlled through parameters present in these operations.

4.3.2.1 GetFeature, GetGmlObject Detail

The level of detail returned by the GetFeature, and GetGmlObject services is controlled by the <*traversexlink*> attribute. It is used to request either just the selected feature type, or the selected and associated feature types.

4.3.2.2 GetObject Detail

The ObjectDetail attribute is used to request either a Full or Summary description of each object.

1. Full Description
Returns all the properties of the objects selected.
2. Summary Description
Returns only a few properties of the objects selected, for the purpose of display, sufficient for the Operator to understand what the object is, plus enough to support identification of object references and refresh.
In addition to the Object Reference properties, specified in Section 4.2.1, the Summary may include the following kinds of properties, dependent on the object type:

- **Location**: the geographic location or extent of the object.
- **Production State**: [Working, Pending, Active, History, Future]
- **Last Modified Date/Time**: date/time when the object was last updated.
- **Effective Start Date**: publication date, if applicable.
- **Effective End Date**: publication date, if applicable object is History.

The Summary properties vary by ObjectType. For inherently "large" objects (e.g., attachments or geospatial reference data) the object metadata is included in the Summary (including the size of the object) and the actual content is retrieved when the Full object is requested.

For a GetObject operation, if the request asks for a Summary description, the server may still return a Full description if it is unable to provide a Summary, or if in the implementation of the server it is decided that the Full description is not large enough to warrant a Summary description. The response message indicates whether the server is returning the Full or the Summary description.

4.3.3 Associations

Associations are essential for browsing or traversing the network of objects that form the aeronautical information space. However, there may be occasions where the client application only wants the properties for a particular ObjectType and does not need to know what other objects are associated.

4.3.3.1 GetFeature, GetGmlObject Associations

The inclusion of associated objects returned by the GetFeature, and GetGmlObject services is controlled by the <***traversexlink***> attribute.

4.3.3.2 GetObject Detail

The level of detail returned by the GetObject service is determined by the AssociatedObjects attribute, with the values True or False. When associated objects are requested, only the Summary descriptions of the associated objects are returned, even if the ObjectDetail parameter indicated Full (in that case the full object details are returned for the primary ObjectType requested, and summary object details are returned for the associated objects).

4.4 Object Model

The Object Model is partitioned into Subject Areas, as described in Table 4.4.1.

Table 4.4.1 Object Model Subject Areas

Subject Area	Description	Applicable Web Service Operations	Specification
Aeronautical Information	Approach, departure and arrival procedures, airspace constructs, ground facilities, obstacles, temporary flight restrictions (e.g., NOTAMs).	GetGmlObject GetFeature PutFeature GetObject PutObject	AIXM 5
Geospatial Reference Information	Information used to visually assist Operator: aeronautical charts, airport layout plans, ground maps, imagery, radar tracks, etc.	GetObject	*Map/image metadata needs to be captured within an ObjectType. Data may be transferred in any number of formats.*
System	System specific configuration data, e.g., available fix names list, code lists, country codes, user account configuration, etc.	GetConfigData PutConfigData	*TBD*

4.4.1 Aeronautical Information

4.4.1.1 Overview

The AIXM-5 XML schema encoding is used to exchange aeronautical information data. The AIXM model focus is on the feature properties and relationships that directly support aircraft navigation in the air and on the ground and thus have relevance to pilots, aircraft and air traffic control. The concepts and design of AIXM summarized in this document, are fully described in AIXM 5 Exchange Model Goals, Requirements and Design [AIXM-5-MGRD], and Data Modeling Conventions for AIXM 5 [AIXM-5-CONV] documents.

The AIXM model is organized around features, and is derived from the OpenGeospatial GML standard. Feature properties can change over time (e.g., a navigation aid, which can have a frequency change). The AIXM conceptual schema is presented in UML, and is implemented in an XML schema, The web service payload is specified through the feature collections exchanged in a GetFeature, GetGMLObjects or PutFeature operation message.

The UML model is composed of classes, relationships (association or aggregation) and properties. An abstract class is shown by an italicized class name. The classes that inherit from the abstract class have a triangle symbol on the association line to the abstract (base) class. Properties describe and characterize the feature/object (e.g., a runway has a property indicating the runway width). Classes are related through association or aggregation. An aggregation implies ownership and coincident lifetime of the parts by the whole.

AIXM includes an extension model that allows AIXM XML documents to contain additional properties that might be specific to a system interface specification or other use case. Extensions allow for additional AIXM Feature and Object properties, and relationships through a standard approach for implementing and documenting extensions.

The FAA has created an extension model for AIXM 5, referred to in this document as the AIXM-5 FAA Profile. This is the model used for the GetGmlObject, GetFeature and PutFeature aeronautical information payload.

The AIXM 5 FAA Profile incorporates extensions originating from:

1. The IPDS Application, captured in the Aeronautical Information Requirements Specification [IPDS-AIRS]. These are extensions for criteria properties, and the minimum set of supplementary properties that IPDS requires to support instrument procedure construction and evaluation.

2. The FAA Enterprise, for the purpose of supporting the flight procedures group and other downstream consumers.

4.4.1.2 Object Versions

Temporality in AIXM data exchange is implemented through a TimeSlice data content model that is defined as part of GML. A TimeSlice encapsulates the time varying properties of a dynamic feature.

The AIXM TimeSlice UML model is illustrated in Figure 4-3. As shown in the model, all AIXM Features derived from a Feature base class that has a static component that contains properties for an artificial identifier. All other properties of the feature are assumed to be temporal. The temporal feature properties are encapsulated into an AIXMTimeSlice object. Each AIXMTimeSlice object contains a valid time interval and an interpretation property. The interpretation property indicates the temporal component that is being modeled. Valid values for the interpretation are:

• Baseline – The state of a feature and all of the feature properties as a result of a permanent change. The Baseline state of a feature also exists when the feature is initially created. The baseline state lasts until the next permanent change.

• Version – The state of a feature and all the feature properties during the time period between two changes.

• Permanent Delta – A set of properties that have changed or will change permanently. The permanent delta will result in a new baseline.

• Temporary Delta – A set of properties that are effective for a limited time. The result is a temporary change to an underlying feature version.

The IFPA web services use Version time slices only. This simplifies the interface and processing of AIXM messages. For example, if there are four versions of the KPHL aerodrome object (one Active, one Pending, and two Working), a request for all Active, Pending, Working KPHL aerodrome objects, would return four AIXMTimeSlice objects, one for each version. Each AIXMTimeSlice object having all the attributes of the KPHL aerodrome for that version.

4.4.1.3 Object Types

Table 4.4.1.3 lists the Aeronautical Information object types that are supported by the GetGmlObject, GetFeature, GetObject, PutFeature, and PutObject services. A '□' indicates that the object type is supported by a service from Airnav.

Table 4.4.1.3 Aeronautical Information Object Types Supported for This Interface Version

Object Type	GetGmlObject	GetFeature	GetObject	PutFeature	PutObject
Aerodrome/ Heliport	✓	✓	N/A	✓	N/A
Runway	✓	✓	N/A	✓	N/A
Communication Service	✓	✓	N/A	Module 2	N/A
Navaid	✓	✓	N/A	✓	N/A
Fix	✓	✓	N/A	N/A	N/A
MSA/ESA	✓	✓	N/A	N/A	N/A
Holding Pattern	✓	✓	N/A	N/A	N/A
Obstacle	✓	✓	N/A	N/A	N/A
Approach Procedure	✓	✓	N/A	N/A	N/A
ESV	✓	✓	N/A	✓	N/A

The AVN-WS allows Attachments to be associated with any AIXM-5 feature.

4.4.1.4 AIXM-5 Payload Schema

The AIXM-5 schema is implemented in XML schemas that define the XML schemas for each of the feature collections to be transferred.

Each feature has common static properties, and an array of time slices. Each time slice has a time interval, interpretation and property group – the property values in effect at the time slice interval. Associations between features are achieved through xlinks.

4.4.1.5 Schema Files

The AIXM-5 schema files include the AIXM-5 .xsd files, FAA extensions to the AIXM-5 schema and included or imported .xsd files that define third party base types (e.g., GML .xsd files) and common types.

4.4.2 System

4.4.2.1 Overview

System-specific configuration data, e.g., country codes, etc. Configuration data is organized into 'configuration sets'.

4.4.2.2 Configuration Sets

Code Lists will be supported by the GetConfigData and PutConfigData services.

Code Lists are lists, that have code types and names. For example, country code list has list of 2-character codes and full country name. The code lists required depends on set of properties being provided for aeronautical information features.

Code List	Values
Country Code	2-character country code Full country name Host Country or Territory
Aerodrome Code	Aerodrome ID code ICAO code Full aerodrome name

4.4.2.3 Object Schema

***TBD**: Configuration Set schema model, to be provided by FAA.*

4.5 Message Pattern

Every message has a header, however, it is slightly different based on whether it is a request or response. All operations include the Request Summary and Response Summary. Data is transferred via the Object Set or as an Attachment.

4.5.1 Request

A request message consists of the following:

- Request Header – mandatory
- Request Summary – optional
- Object Set – only used for Put operations

4.5.1.1 Request Header

Purpose
The Request Header serves to uniquely identify the request, its sender, and to establish the correct XML Schema for its interpretation, i.e., the InterfaceVersion.

Content
The Request Header is standard across all requests and contains the following mandatory elements:

• **InterfaceVersion:** identifies the major and minor (M.N) web services interface version.

• **UserAgent:** identifies client machine and application software from which the request originated.

• **UserID:** User's unique ID within the realm of the Data Source.

• **SessionToken:** an obfuscated or encrypted string provided by Data Source on initialization of connection. This element is optional for the ServerStatus operation, enabling the client application to determine whether or not connectivity with the Data Source exists (typically while configuring) prior to obtaining a SessionToken. The ValidateUser operation generates the SessionToken, hence it is not an input. All other operations require the SessionToken.

• **RequestID:** identifier provided by the client application that serves to uniquely identify a service request.

Usage
The client application builds the Request Header, supplying the SessionToken obtained from the Data Source, if required.

The server verifies that the SessionToken is valid, and will not satisfy the request if invalid, instead a Fault Message is returned. A valid SessionToken may expire after a pre-determined period of time, at the discretion of the Data Source.

Example
• InterfaceVersion: 2.1
• UserAgent: NFPG21 Windows XP 1.0.3 TARGETS 1.3
• UserID: Glenn
• SessionToken: !@#$%^&*()
• RequestID: 324

4.5.2 Response

A response message consists of the following:

- Response Header – mandatory
- Response Summary – optional
- Object Set – optional

4.5.2.1 Response Header

Purpose
The Response Header serves to uniquely identify the Response, linking it to the original Request.

Content
The Response Header is standard across all responses and contains the following mandatory elements:

- **InterfaceVersion:** identifies the major and minor (M.N) web services interface version.
- **UserAgent:** identifies client machine and application software from which the request originated.
- **UserID:** User's unique ID within the realm of the Data Source.
- **RequestID:** identifier provided by the client application that serves to uniquely identify a service request.
- **Server:** identifies server machine and server-side software that generated the response.

Usage
The server application builds the Response Header, largely by echoing back the elements of the Request Header.

Example

- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.0.3 Targets 1.3
- UserID: Glenn
- RequestID: 324
- Server: avnokcprd.amc.faa.gov Hub 2.1

4.5.3 Fault

A fault message consists of the following:
• Fault Description – mandatory

The WS-I Basic Profile specifies a standard Fault response as follows:

- **FaultCode:** indicates one of the standard SOAP fault codes [SOAP]. In fact, it appears that this code would always be set to "Sender".
- **FaultString:** human-readable indication of the fault.
- **FaultActor:** indicates who caused the fault.
- **Detail:** contains custom Fault message.

4.5.3.1 Fault Description

The fault description element contains one top level element, **ServiceException**, that has been declared under the [http://avn.ato.faa.gov/ws/faults/] namespace. This element contains detailed information on the exception as well as the route taken through the Enterprise that caused this exception to be raised.

Table 4.5.3.1 ServiceException Element

ServiceException				
Attributes	**Required?**		**Type**	**Description**
No Attributes				
Elements	**Min**	**Max**	**Type**	**Description**
ExceptionInformation	1	1	ExceptionInformationType	Information on the exception that has occurred.
ServiceRouteInformation	1	1	ServiceRouteInformationType	Information regarding the call stack through the service layers.
ExceptionInformationType				
Attributes	**Required?**		**Type**	**Description**
Type	Yes		ExceptionTypeType	The type of exception that has occurred.
Severity	Yes		ExceptionSeverityType	The severity of the exception.

Timestamp	Yes	UTC Time	The time of the exception.
ID	No	String data	A uniquely identifiable ID, created by the source of the exception, for correlating the client side with the server side exception at a later date.

Elements	**Min**	**Max**	**Type**	**Description**
Summary	1	1	String data	The summary description of the exception. Human readable. The target is the end user of the system.
Description	1	1	String data	A human readable description of the exception. Gives details as to the cause of the exception and possible recourse. The target is the end user of the system..
Log	0	1	String data	The machine log raised by the source of the exception. Not for end user consumption.

ServiceRouteInformationType

Attributes	**Required?**	**Type**	**Description**
No Attributes			

Elements	**Min**	**Max**	**Type**	**Description**
ServiceLayer	1	inf	ServiceLayerType	One for each particular point along which the web service call may be traced.

ServiceLayerType

Attributes	**Required?**	**Type**	**Description**
ServiceName	Yes	String data	Web Services service name being called. Directly from the WSDL document.
OperationName	Yes	String data	Web Services operation name being called. Directly from the WSDL document.
InterfaceVersion	Yes	String data	The version of the interface currently in use.
isFaultInstigator	Yes	Boolean *Default Value*: FALSE	Set to true if the current ServiceLayer raised the exception originally.

MachineName	No		String data	The name of the machine that raised exception occurred on.
MachineAddress	No		String data	The address of the machine that raised the exception.
ServiceAdminContact	No		String data	The administration contact for the machine that raised the exception.
Elements	**Min**	**Max**	**Type**	**Description**
RequestMessage	1	1	String data	The original Request Message to the Operation being called at this service layer.

4.5.3.2 Fault Exception Types

The general exception types, which are common to all services, are listed in the Table 4-10 below complete with description and possible recourse.

Table 4.5.3.2 Exception Types

Enumeration	Description	Recourse
APPLICATION_EXCEPTION	An exception has occurred in the application.	Notify help desk.
DATA_SOURCE_UNAVAILABLE	The data source has timed out or is otherwise unavailable.	Retry request after a delay.
DATA_STALE	The data being acted upon is stale and thus must be refreshed.	Refresh data and repeat operation.
DATA_UNAVAILABLE	The data being acted upon is unavailable.	Notify user of problem.
DATA_VALIDATION_ERROR	The data submitted is not valid.	Notify user of validation problems, fix, then resubmit.
INVALID_PERMISSIONS	The client does not have the appropriate permissions to perform the given operation.	Notify user of permission problems.
INVALID_VERSION	The version of the web service being called is old/invalid.	Notify help desk.
NOT_AUTHENTICATED	The client is not authenticated with the enterprise.	Re-authenticate client.
QUERY_SIZE	The size of the query is too large to handle a must be redone with a smaller request.	Repeat operation with smaller query size.

SCHEMA_VALIDATION_ERROR	A message failed validation.	Notify help desk.
UNKNOWN	Umbrella fault type for as yet unknown exceptions.	Gain insight from severity as to recourse.

The severity types for exceptions are listed in the table below.

Table 4.5.3.3 Exception Severity Types

Enumeration	**Description**
RETRY	Cue that the exception is not fatal and that a simple retry will possibly remedy the situation.
MINOR	Cue that a recourse is possibly available and notification to user plus logging may not be necessary.
MAJOR	Cue that a recourse is not possible and notification to the user plus logging may be necessary. Also notification to the Help Desk may be necessary.

5 Filters

This section describes the use of the OpenGIS® Filter Encoding Implementation Specification [FILTER] to define filters to limit object queries.

This section follows the same order of description as in the OpenGIS® Filter Encoding Implementation Specification, and includes parts of that specification.

A filter expression is used to select feature instances to be returned (for example, a bounding box). Filters are as specified OpenGIS® Filter Encoding Implementation Specification**, with the limitations described below.**

The Filter Encoding Implementation schemas may be obtained at
http://schemas.opengeospatial.net/filter/1.1.0.

The filter encoding is used in a GetFeature and GetObject operations to define query constraints.

5.1 Property names

The **<PropertyName>** element is used to encode the name of any property of an object. The property name can be used in scalar or spatial expressions to represent the value of that property for a particular instance of an object.

For the AVN-WS GetFeature service, only the following property names in Table 5-1 are allowed.

Table 5.1 GetFeature Allowed Property Names

Property Name	Usage
country	2 character Country code
state	2 character State/Province code
objectName	a humanly recognizable term for the object. This property may be filtered with the PropertyIsLike comparison operator, It must include at least three (3) alphanumeric characters. Wildcarding symbols will follow SQL conventions: '_' for single character, '%' for any number of characters.
aixm:identifier	The objectID aixm:identifier (which is used to assign a globally unique identifier to the same conceptual feature object).
objectVersion	Version number of a feature
productionState	any combination of the set of [Working, Pending, Active, History, Future].
beginPosition	Effective date (in GML 3.1.1 format)
endPosition	End date
lastModifiedPosition	Date feature was last modified.

For the AVN-WS GetObject service, only the following property names in Table 5-2 are allowed.

Table 5.1.2 GetObject Allowed Property Names

Property Name	Usage
lastModifiedPosition	Date object was last modified.

5.2 Property References ([FILTER])

The filter property names (in Table 5.1 above) do not reference specific elements or attributes of elements. As objects can also include complex or aggregate non-geometric properties a filter expression processor must use the subset of XPath expressions defined in [FILTER] in order to unambiguously reference simple properties and the properties and sub-properties of objects with complex or aggregate properties or properties encoded as XML attributes.

For the AVN-WS, the properties, other than objectName, are common to all features. The actual attribute being filtered when objectName is given, depends on the feature type, as listed in Table 5-4.

Table 5.2 Object Name References

Feature	Object Name Element
aixm:AerodromeHeliport	designator
aixm:Runway	designator
aixm:ServiceAtAerodromeHeliport	Cannot be filtered by Object Name.
aixm:Navaid	identifier
aixm:DesignatedPoint	designator
aixm:Holding	Cannot be filtered by Object Name.
aixm:InstrumentApproachProcedure	name

5.3 Filter

A filter is any valid predicate expression that can be formed using the elements defined in the OpenGIS® Filter Encoding Implementation Specification [FILTER]. The root element **<Filter>** contains the expression which is created by combining these elements.

The root element of a filter expression, <**Filter**>, is defined by the following XML Schema fragment:

```
<xsd:element name="Filter" type="ogc:FilterType"/>
        <xsd:complexType name="FilterType">
            <xsd:choice>
                 <xsd:element ref="ogc:spatialOps"/>
                 <xsd:element ref="ogc:comparisonOps"/>
                 <xsd:element ref="ogc:logicOps"/>
                 <xsd:element ref="ogc:_Id" maxOccurs="unbounded"/>
            </xsd:choice>
        </xsd:complexType>
```

The elements <**logicalOps**>, <**comparisonOps**> and <**spatialOps**> are substitution groups for logical, spatial and comparison operators. Logical operators may be used to combine spatial operators and comparison operators in one filter expression. The <**_Id**> element is the head of a substitution group for object identifiers.

5.3.1 Spatial Operators

A spatial operator determines whether its geometric arguments satisfy the stated spatial relationship. The operator evaluates to TRUE if the spatial relationship is satisfied. Otherwise the operator evaluates to FALSE.

Only the **<BBOX>** element which defines a bounding box constraint based on the **gml:Envelope** geometry is supported in AVN-WS. The **<BBOX>** operator should identify all geometries that spatially interact with the box. The calling service must determine which spatial property is the spatial key and apply the BBOX operator accordingly.

5.3.2 SRS Handling Of Literal Geometries In Filter Expressions

AVN-WS does not transform the **gml:Envelope**. co-ordinates (the Spatial Reference System (SRS) of the **gml:Envelope** will be ignored). Spatial queries are performed using the co-ordinates as supplied. Matching features are also returned in the coordinate reference system with which they are stored.

5.3.3 Comparison Operators

A comparison operator is used to form expressions that evaluate the mathematical comparison between two arguments. If the arguments satisfy the comparison then the expression evaluates to TRUE. Otherwise the expression evaluates to FALSE.

This type definition includes the **matchCase** attribute which is of type Boolean and controls whether string comparisons are caseless or not. A value of true means that string comparisons also match case. This is the default value. A value of false means that string comparisons are performed caselessly.

In addition to the standard set of comparison operators, this specification defines the elements **<PropertyIsLike>**, **<PropertyIsBetween>** and **<PropertyIsNull>**. The **<PropertyIsLike>** element is intended to encode a character string comparison operator with pattern matching. The pattern is defined by a combination of regular characters, the **wildCard** character, the **singleChar** character, and the **escapeChar** character. The **wildCard** character matches zero or more characters. The **singleChar** character matches exactly one character. The **escapeChar** character is used to escape the meaning of the **wildCard**, **singleChar** and **escapeChar** itself.

The **<PropertyIsNull>** element encodes an operator that checks to see if the value of its content is NULL. A NULL is equivalent to no value present. The value 0 is a valid value and is not considered NULL.

The **<PropertyIsBetween>** element is defined as a compact way of encoding a range check. The lower and upper boundary values are inclusive.

5.3.4 Logical Operators

A logical operator can be used to combine one or more conditional expressions. The logical operator AND evaluates to TRUE if all the combined expressions evaluate to TRUE. The operator OR operator evaluates to TRUE is any of the combined expressions evaluate to TRUE. The NOT operator reverses the logical value of an expression.

The elements <**And**>, <**Or**> and <**Not**> can be used to combine scalar, spatial and other logical expressions to form more complex compound expressions.

5.3.5 Object Identifiers ([FILTER] Section 11)

A object identifier is meant to represent a unique identifier for an object instance within the context of the web service that is serving the object. This OpenGIS® Filter Encoding Implementation Specification does not define a specific element for identifying objects but instead defines the abstract element <**_Id**> as the head of an XML substitution group that may be used to define an object identifier element for specific object types.

For GetFeatures, the AVN-WS uses the aixm:identifier (which is used to assign a globally unique identifier to the same conceptual feature object. The codespace attribute is used to identify the maintaining authority or scheme for the identifier, e.g., UUID).

5.3.6 Expressions

An expression is a combination of one or more symbols that evaluate to single Boolean value of true or false. Expressions are not supported in AVN-WS.

5.3.7 Arithmetic Operators

Arithmetic operators are binary operators meaning that they accept two arguments and evaluate to a single result. Arithmetic operators are not supported in AVN-WS.

5.3.8 Literals

A literal value is any part of a statement or expression that is to be used exactly as it is specified, rather than as a variable or other element. The **<Literal>** element is used to encode literal scalar and geometric values. Literal geometric values must be encoded as the content of the **<Literal>** element, according to the schemas of GML as described in the Geography Markup Language (GML) Implementation Specification. That is to say that any literal GML geometry instance must validate against the XML Schemas defined for GML3.

5.3.9 Functions

A function is a named procedure that performs a distinct computation. A function may accept zero or more arguments as input and generates a single result. Functions are not supported in AVN-WS.

5.3.10 Filter Capabilities

The filterCapabilities.xsd schema defines a capabilities document section that shall be instantiated in the capabilities document of services that use filter encoding. The filter capabilities document section describes what specific filter capabilities are supported by a service. For example, a web feature service that uses filter encoding would include this fragment in its capabilities document to advertise what filter capabilities it supports. Filter capabilities are not supported in AVN-WS.

5.4 Examples

Example 1

A filter checking for features in Washington State.

```
<Filter>
        <PropertyIsEqualTo matchCase=False>
        <PropertyName>state</PropertyName>
        <Literal>WA</Literal>
        </PropertyIsEqualTo>
</Filter>
```

Example 2

Finding all features that have a geometry within the specified bounding box.

```
<Filter>
<BBOX>
        <gml:Envelope srsName="WGS84">
        <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
        <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
        </gml:Envelope>
```

```
</BBOX>
</Filter>
```

Example 3
Features that are Active or Pending inside a Bounding Box.

```
<Filter>
<And>
<Or>
<PropertyIsEqualTo>
        <PropertyName> productionState </PropertyName>
        <Literal>A</Literal>
</PropertyIsEqualTo>
<PropertyIsEqualTo>
        <PropertyName> productionState </PropertyName>
        <Literal>P</Literal>
</PropertyIsEqualTo>
</Or>
<BBOX>
        <gml:Envelope srsName="WGS84">
        <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
        <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
        </gml:Envelope>
</BBOX>
</And>
</Filter>
```

Example 4
All versions of a particular feature

```
<Filter>
        <PropertyIsEqualTo>
                <PropertyName>objectId</PropertyName>
                <Literal>1234</Literal>
        </PropertyIsEqualTo>
</Filter>
```

Example 5
Features that are names "JF%" inside a Bounding Box.

```
<Filter>
<And>
        <PropertyIsLike wildCard="%" singleChar="_" escapeChar="!">
        <PropertyName>objectName</PropertyName>
        <Literal>JF%</Literal>
        </PropertyIsLike>
        <BBOX>
                <gml:Envelope srsName="WGS84">
                <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
                <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
                </gml:Envelope>
        </BBOX>
</And>
</Filter>
```

Example 6
Any object version effective (published) on a specified date (2001-01-15, 20:07:48.11).

```
<Filter>
<And>
        <PropertyIsLessThanOrEqualTo>
        <PropertyName>beginPosition</PropertyName>
        <Literal>2001-01-15T20:07:48.11</Literal>
        <PropertyIsLessThanOrEqualTo>
        <PropertyIsGreaterThanOrEqualTo>
        <PropertyName>endPosition</PropertyName>
        <Literal>2001-01-15T20:07:48.11</Literal>
        <PropertyIsGreaterThanOrEqualTo>
</And>
</Filter>
```

Example 7
Any object version with an effective date within the time period, (2001-01-15 20:07:48.11 to 2001-04-15 12:22:25.30).

```
<Filter>
        <PropertyIsBetween>
        <PropertyName>beginPosition</PropertyName>
        <LowerBoundary>
        <Literal>2001-01-15T20:07:48.11</Literal>
        </LowerBoundary>
        <UpperBoundary>
        <Literal>2001-04-15T12:22:25.30</Literal>
        </UpperBoundary>
        </PropertyIsBetween>
</Filter>
```

Example 8
Any object version with a modified date is greater (later) than or equal to this date/time.

```
<Filter>
        <PropertyIsGreaterThanOrEqualTo>
        <PropertyName>lastModifiedPosition</PropertyName>
        <Literal>2001-01-15T20:07:48.11</Literal>
        <PropertyIsGreaterThanOrEqualTo>
</Filter>
```

6 GetFeature Operation

This section specifies the GetFeature operation: its purpose, design, content, and usage, providing examples.

6.1 Purpose

The GetFeature operation is used to retrieve /refresh a set of aeronautical feature instances from a Data Server using spatial, temporal and textual filter parameters.

6.2 Design

6.2.1 GetFeature (OGC WFS 9.0)

The GetFeature web service request is based on the OpenGeospatial WFS specification. [WFS] defines the schema for requesting and responding to GML features. This specification was selected as it provides standardized service interface to GML features, and as AIXM-5 implements a profile of GML.

The implementation is based on SOAP.

This operation is intended to support general querying/browsing of aeronautical objects in a Data Source.

Design decisions:

1. Although the GetFeature service allows objects of more than one feature type to be requested. Clients should consider requesting only one feature type, to prevent queries against multiple databases that would slow down the response time.

2. Spatial filters can be expressed as a location code (country and state) in order to retain IAPA database inquiry capability, as well as expressed as a bounding box.

3. Wildcarding on object properties is included to match IFPA database inquiry capability.

6.2.2 DescribeFeatureType Operation (OGC WFS 8.0)

The function of the **DescribeFeatureType** operation is to generate a schema description of feature types serviced by a WFS implementation.

6.3 Content

This section tailors the OGC Web Feature Service Implementation Specification [WFS] document sections. It follows the same section ordering, and includes the applicable sections as required.

6.3.1 Common Elements (OGC WFS 7.0)

6.3.1.1 Feature and Element Identifiers (OGC WFS 7.1)

AVN-WS, many instances (versions) of the same feature can exist, for example, Active and Pending versions of a runway. In AIXM-5, the aixm:identifier uniquely identifies a feature, but not a feature instance. Particular instances of a feature are identified through timeslices. For uniqueness, AXIM-5 is extended to include a versionNumber element on a timeslice, which when combined with aixm:identifier identifies a feature instance. The aixm:identifier is set to the control number of the feature, with an appropriate codeSpace.

6.3.1.2 Feature State (OGC WFS 7.2)

A client application uses the application schema definition of a feature type to refer to feature instances of that feature type, and to refer to the names and types of all the properties of those feature instances. The values of all properties of a feature instance constitute the state of that feature instance. A client application references feature instances by the name of their feature type and the names and values of feature properties.

6.3.1.3 Property Names (OGC WFS 7.3)

The property names used must be valid element and attribute names. In addition, property names may be namespace qualified.

6.3.1.4 Property References (OGC WFS 7.4)

The [WFS] states a WFS **must** use XPath expressions, for referencing the properties and components of the value of properties of a feature encoded as XML elements or attributes. It also mandates a subset of the XPath language that must be supported (it does not require a WFS to support the full Xpath language).

6.3.1.5 <Native> Element (OGC WFS 7.5)

The <Native> element is intended to allow access to vendor specific capabilities of any particular web feature server or datastore.

The <Native> element simply contains the vendor specific command or operation.

The AVN-WS does not support <Native> commands/operations.

6.3.1.6 Filter (OGC WFS 7.6)

A filter is used to define the collection of feature instances that are to be returned. Filter specifications are encoded, and are based on the OGC Filter Encoding Implementation Specification [FILTER].

6.3.2 Exception reporting (OGC WFS 7.7)

In the event that a web feature service encounters an error while processing a request or receives an unrecognized request, it generates an XML document indicating that an error has occurred.

6.3.2.1 Common XML Attributes (OGC WFS 7.8)

6.3.2.1.1 Version Attribute

All XML encoded WFS requests include an attribute called version. The mandatory version attribute is used to indicate to which version of the WFS specification the request encoding conforms and is used in version negotiation.

For AVN WS the version reflects the OGC WFS version (currently 1.1.0).
Note, the AVN WS also has a version number that is sent with each message.

6.3.2.2 Service Attribute

All XML encoded WFS requests include an attribute called service. The mandatory service attribute is used to indicate which of the available service types, at a particular service instance, is being invoked. When invoking a web feature service, the value of the service attribute shall be WFS.

6.3.2.3 Handle Attribute

The purpose of the handle attribute is to allow a client application to associate a mnemonic name with a request for error handling purposes. If a handle is specified, and an exception is encountered, a Web Feature Service may use the handle to identify the offending element.

For AVN-WS, clients may use handles, and if supplied, the AVN-WS includes handle attribute in fault responses. The AVN-WS does not make any assumptions about the content of a Handle, clients may define their own content as to what they contain.

6.4 GetFeature Request Message (OGC WFS 9.0)

The Request Message contains the following XML elements:

- **Request Header**
- **Request Summary: contains**
  - **DataSourceNamespace**
  - **GetFeature Request**

6.4.1 GetFeature Request Introduction (OGC WFS 9.1)

The **GetFeature** operation allows retrieval of features from the AVN-WS. A **GetFeature** request is processed by the AVN-WS and when the value of the **outputFormat** attribute is set to **text/gml; subtype=gml/3.1.1**, a GML instance document, containing the result set, is returned to the client.

If a web feature service implements Xlink traversal, a client can use the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes to request that nested property XLink linking element locator attribute (href) XLinks are traversed and resolved if possible.

6.4.2 Request (OGC WFS 9.2)

The **<GetFeature>** element contains one or more **<Query>** elements, each of which in turn contains the description of a query. The results of all queries contained in a **GetFeature** request are concatenated to produce the result set.

The optional **outputFormat** attribute specifies the format of the response to a **GetFeature** request. The default value is **text/xml; subtype=gml/3.1.1,** and this indicates that a valid GML3 document, that validates against a valid application schema be returned. This default will not be supported at this time. As AIXM-5 has a GML profile derived from GML 3.1.1, the AVN WS returns an AIXM-5 Feature Collection.

Table 6.5.2 lists the possible values for the **outputFormat** attribute:

Table 6.4.2 Values for outputFormat Attribute

outputFormat Value	Description
text/xml; subtype=gml/3.1.1	This value indicates that an XML instance document must be generated that validates against an AIXM-5 (and GML3) application schema. This is the default values of the outputFormat attribute if the attribute is not specified in the GetFeature request. Note: This will not be supported at this time.
application/x-gzip; subtype=aixm/5rc2	This value indicates that a compressed XML instance document must be generated that validates against an AIXM-5 RC2 application schema. See Section 3.7 on compression.

A web feature service may respond to a **GetFeature** request in one of two ways. It can either generate a complete response document or it may simply return a count of the number of features that a **GetFeature** request would return. The optional **resultType** attribute is used to control how a web feature service responds to a **GetFeature** request.

Table 6.5.2.1 Values for resultType Attribute

resultType Value	Description
Results	The default value results indicates that a web feature service should generate a complete response that contains all the features that satisfy the request..
Hits	The value hits indicates that a web feature service should process the GetFeature request and rather than return the entire result set, it should simply indicate the number of feature instance of the requested feature type(s) that satisfy the request. That is that the count should only include instances of feature types specified in the typeName attribute (i.e. GetFeature/Query/@typeName).

The optional **maxFeatures** attribute can be used to limit the number of explicitly requested features (i.e. features specified via the GetFeature/Query/@typeName attribute) that a **GetFeature** request presents in the response document. The maxFeatures value applies to whole result set and the constraint is applied to the features in the order in which they are presented. In addition, feature members contained in a requested feature collection do not count – the requested feature collection counts as one feature. Once the **maxFeatures** limit is reached, the result set is truncated at that point. There is no default value defined and the absence of the attribute means that all feature type instances in the result should be returned to the client.

For AVN-WS, the **maxFeatures** attribute is be supported. The AVN-WS may still report an exception for queries which are not sufficiently constrained.
Each individual query packaged in a **GetFeature** request is defined using the **<Query>** element. The **<Query>** element defines which feature type to query, what properties to retrieve and what constraints (spatial and non-spatial) to apply to the feature properties in order to select the valid feature set.

Note, in order to reduce message query processing times, and message sizes, the IFPS applications only request one feature type to query, however the GetFeature request does allow multiple queries to be included in a request.

The mandatory **typeName** attribute is used to indicate the name of one or more feature type instances to be queried. Its value is a list of namespace-qualified names (XML Schema type QName, e.g., aixm:Runway).

Note, in [WFS] feature types that are supported are advertised in the Capabilities document of the WFS. For AVN-WS in Module 1, the Capabilities document of the WFS is not supported. The advertised allowed feature types are listed in Table 10-3.

The [WFS] specifies that more than one **typename** attribute indicates that a join operation is being performed. For AVN-WS, this is not supported. (It is not clear how joins work in the WFS, and it appears that this option may be dropped from the next release of [WFS].)

For each feature type, the additional feature types that may be requested with the feature are listed.

Table 6.4.2.2 Allowed typeName

Allowed typeName's
aixm:AerodromeHeliport
aixm:Runway
aixm:ServiceAtAerodromeHeliport
aixm:Navaid
aixm:DesignatedPoint

aixm:Holding
aixm:InstrumentApproachProcedure

The optional **featureVersion** attribute on the <**Query**> element is included in order to accommodate systems that support feature versioning.

Note: As a result of comments from ISO, feature versioning has been removed from the next version of WFS (currently called V1.2) of the WFS specification (it appears that there was not enough detail in the specification to allow an interoperable implementation to be written).

For AVN-WS the **featureVersion** attribute is not supported. Selection of specific versions of a feature are achieved through the <Filter> element.
The optional **srsName** attribute of the <**Query**> element is used to specify a specific WFS-supported SRS to be used for returned feature geometries.

For AVN-WS, the **srsName** attribute is not required, and is ignored. Features are returned with the spatial properties, with which it is defined. A feature collection may have a mixed set of vertical and horizontal datum.

The optional **traverseXlinkDepth** attribute indicates the depth to which nested property XLink linking element locator attribute (href) XLinks in all properties of the selected feature(s) are traversed and resolved if possible. A value of "1" indicates that one linking element locator attribute (href) XLink is traversed and the referenced element returned if possible, but nested property XLink linking element locator attribute (href) XLinks in the returned element are not traversed. A value of "*" indicates that all nested property XLink linking element locator attribute (href) XLinks are traversed and the referenced elements returned if possible. The range of valid values for this attribute consists of positive integers plus "*".

The **traverseXlinkExpiry** attribute is specified in minutes. It indicates how long a Web Feature Service should wait to receive a response to a nested **GetGmlObject** request.

For AVN-WS, the **traverseXlinkExpiry** attribute is not supported.

For AVN-WS, the **traverseXlinkDepth** is supported to indicate that xlinks get traversed (and so referenced features are included in the response). This provides a mechanism for a client to request summary and full details on features.
The traversal also traverses 'referenced by' associations. For example, a request for Runway with **traverseXlinkDepth = 1,** would also return RunwayDirection features that are associated to the Runway (as Runway is referenced by RunwayDirection in AIXM-5).

For AVN-WS, the **traverseXlinkDepth** attribute is interpreted as follows:

- **traverseXlinkDepth = 0** - the AVN-WS returns the requested feature type only (and any objects that are an aggregate of that feature). This is equivalent to a "Summary" request/response.

- **traverseXlinkDepth = *** - the AVN-WS will returns the requested feature type, and all associated feature types, as shown in Table 10-4 below. In effect xlinks are traversed up to certain points, depending on the requested feature type. This is equivalent to a "Full" request/response.

The value of each <**wfs:PropertyName**> element is a namespace-qualified name (XML Schema type QName, e.g. myns:address) whose value must match the name of one of the property elements in the GML representation of the relevant feature. The relevant feature is of the type indicated as the value of the **typeName** attribute of the parent <**Query**> element. If <**wfs:PropertyName**> is absent all properties are returned. If <**wfs:PropertyName**> is present then selected plus any 'mandatory' properties are returned.

Note: In AIXM, all property elements are optional, so there are no mandatory elements in a feature.

A <**wfs:XlinkPropertyName**> element may be used instead of the <**wfs:PropertyName**> element to enumerate feature properties that should be selected during a query and whose values should be processed by a **GetGmlObject** operation as specified by the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes to traverse and resolve nested property XLink linking element locator attribute (href) XLinks that those values contain. The **traverseXlinkDepth** and **traverseXlinkExpiry** attributes on the <**wfs:XlinkPropertyName**> element apply only to the value of the named property, and override the values of the **traverseXlinkDepth** and **traverseXlinkExpiry** attributes, if any, on the <**GetFeature**> element.

The <**Filter**> element is used to define constraints on a query. Both spatial and/or nonspatial constraints can be specified as described in Section 5.3.
A <**Filter**> element is required. It must contain either a spatial (bounding box), or country/state code properties in order to constrain the feature collection to be retrieved.

The <**SortBy**> element is used to specify a list of property names whose values should be used to order (upon presentation) the set of feature instances that satisfy the query.

Table 6-4 enumerates the AIXM-5 feature types that are returned if traverseXlinkDepth = *. Feature types marked as 'Local to XML Document' have no aixm:identifier (FAA Enterprise control number), and can only be referenced by other features in the same XML document. It also lists the AIXM-5 objects that are aggregates of each feature type.

Table 6.4.2.3 Feature Collections

Feature Type Requested	AIXM Feature Types in the Feature Collection	Local to XML Docum en t	AIXM Objects Included with the Feature	Comments
AerodromeHeliport				
	AerodromeHeliport		ElevatedPoint (hasReferencePoint)	
Runway				
	Runway		SurfaceCharacteristics	The Runway feature has an Object Reference (association) to an Aerodrome Heliport feature.
	RunwayDirection		RunwayCenterlinePointR ole (hasPositions)	The Runway Direction objects that ar associated to the returned Runway objects. The Runway Direction featu has an Object Reference (association) to a Runway feature. The Runway Direction feature has an Object Reference (association) to Runway Centerline Point features.
	RunwayCenterlinePoint	🗑	ElevatedPoint (locatedAt)	The Runway Centerline Point objects that are associated to the returned Runway Direction objects.
	RunwayDeclaredDistance	🗑		The Runway Direction Declared Distance objects that are associated to the returned Runway Direction object The Runway Direction Declared Distance feature has an Object Reference (association) to a Runway Direction feature.
	RunwayProtectArea	🗑		Runway Protect Area derives from th Landing Protection Area abstract feature. The Runway Protect Area objects that are associated to the returned Runway Direction objects. T Runway Protect Area feature has an Object Reference (association) to a Runway Direction feature.
	RunwayVisualRange	🗑		The Runway Visual Range objects th are associated to the returned Runway Direction objects. The Runway Visua Range feature has an Object Referenc (association) to a Runway Direction feature.

	VisualGlideSlopeIndicator			The Visual Glide Slope Indicator objects that are associated to the returned Runway Direction objects. T Visual Glide Slope Indicator feature h an Object Reference (association) to a Runway Direction feature.
	RunwayDirectionLightSyste m			Runway Direction Light System deriv from the Surface Light System abstra feature. The Runway Direction Light System objects that are associated to t returned Runway Direction objects. T Runway Direction Light System featu has an Object Reference (association) to a Runway Direction feature.
	ApproachLightingSystem			The Approach Lighting System objec that are associated to the returned Runway Direction objects. The Approach Lighting System feature ha: an Object Reference (association) to a Runway Direction feature.
ServiceAtAerodromeHeliport				
	ServiceAtAerodromeHeliport		ElevatedPoint (locatedAt) Timetable (hoursOf)	Service At Aerodrome Heliport derive from the Service abstract feature.
	Frequency	🗑	CallsignDetail (identifiedBy)	The Frequency objects that are associated to the returned Service objects. The Frequency feature has an Object Reference (association) to a Service feature.
Navaid				
	Navaid		NavaidComposition ElevatedPoint (hasNavigableLocation)	
	NavaidEquipment		ElevatedPoint (hasLocation)	The Navaid Equipment objects that ar associated to the returned Navaid objects. Navaid Equipment includes a derived classes (VOR, NDB, etc).
	RadioFrequencyLimitation		CircleSector (concerns)	The Radio Frequency Limitation objects that are associated to the returned Navaid Equipment objects. The Radio Frequency Limitation feature has an Object Reference (association) to a Navaid Equipment feature.

DesignatedPoint				
	DesignatedPoint		Point (locatedAt)	Only returns from FIX database.
	PointReference		AngleIndication (basedOnAngle) DistanceIndication (basedOnDistance)	The PointReference features that are associated to the Designated Point. These are the Describing facilities fro FAA Enterprise Fix application.

6.4.3 Response Message (OGC WFS 9.3)

The Response Message contains the following XML elements:

- **Response Header:** as specified in Section 4.5.2.1.
- **Response Summary:** contains
    - **DataSourceNamespace:** identifies the Data Source uniquely.
    - **Object Set:** contains the response objects

If the **outputFormat** attribute is **application/x-gzip; subtype=aixm/rc2,** each element in the object set may contain XML data in an ASCII encoded binary form (XML Schema *base64binary*) and is optionally compressed. The response element also contains the truth-value of **compression** with a *false* denoting the response data is not compressed and a *true* denoting compression has occurred on the data.

Optionally, **timestamp** and **numberOfFeatures** attributes may also be added to the response object for the client applications' use. The mime-type of the response data is controlled by the **outputformat** attribute of the request message. The uncompressed data attribute equates to an XML document with root aixm:AbstractFeatureMemberArrayType.

Note, in [WFS] the wfs:FeatureCollectionType base class is gml:AbstractFeatureCollectionType. To align with AIXM-5 (which defines base schemas that can be more easily versioned to GML v3.2), and to require all possible feature types to be included in the schema, the wfs:FeatureCollectionType base class is changed to aixm:AbstractFeatureMemberArrayType.

Note, the [WFS] specifies that if there are traversed Xlinks in the response, then a comment containing the locator attribute followed by the child elements contained in the XLink linking element, are written to the response element. Effectively, the referenced features are nested in the referencing feature. As this is not consistent with AIXM-5 feature collections, for AVN-WS all referenced features are part of the feature collection (a feature instance is never nested in another feature instance).

The content of the **<wfs:FeatureCollection>** element is controlled by the value of the **resultType** attribute on the **<GetFeature>** element which is described above. If the specified value of the **resultType** attribute is **results** (the default value) then the AVNWS generates a complete response as the content of the **<wfs:FeatureCollection>** element.

If, however, the value of the **resultType** attribute is specified as **hits**, the AVN-WS service generates a **<wfs:FeatureCollection>** element with no content (i.e. empty) and populate the values of the **timeStamp** attribute and the **numberOfFeatures** attribute. In this way a client may obtain a count of the number of features that a query would return without having to incur the cost of transmitting the entire result set.

For AVN-WS the **lockId** attribute is not used. The optional **timeStamp** attribute is used by AVN-WS to indicate the time and date when a response was generated.
The optional **numberOfFeatures** attribute is used by AVN-WS to indicate the number of features that are in the response document. The count should only include feature type instances of the feature type name specified in the **typeName** attribute of the **<Query>** element (i.e. GetFeature/Query/@typeName) used to generate the response.

Any GML document generated by a WFS implementation, in response to a query where the **outputFormat** is the MIME type **text/xml; subtype=gml/3.1.1**, must reference an appropriate GML application schema document so that the output can be validated.

This may be accomplished using the **schemaLocation** attribute. This attribute indicates the physical location of one or more schema documents which may be used for local validation and schema-validity assessment. The **schemaLocation** attribute value contains pairs of values. The first member of each pair is the namespace for which the second member is the hint describing where to find to an appropriate schema document.

The physical location of the schema documents is specified using a URI.

For AVN-WS the **schemaLocation** attribute is not supported.
The response Object Set should also denote the source application. This allows the *Enterprise Service Bus* to aggregate data from multiple providers into a single response message if necessary.

6.4.4 Fault Message (OGC WFS 9.4)

The Fault Message contains the following XML elements:

- **Fault Description:** as specified in Section 4.5.3.1
- **Fault Exceptions:** as specified in Section 4.5.3.2

6.5 Usage

Scenarios where IFPA Applications use GetFeature are as follows:

1. Retrieving objects within a geographic area (populating a Work Area)
Requires identifying objects within a spatial bounding box.

2. Refreshing a Work Area
Requires identifying objects that have been modified since a particular date/time.

3. Retrieving objects within a geographic area matching a name
Requires wildcarding of an ObjectName.

4. Retrieving objects within a geographic area using a combination of text filter parameters.

6.6 Examples

6.6.1 Retrieving Objects Within a Geographic Area

Obtaining a summary of the Aerodromes within a spatial bounding box.

GetFeature Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 Targets
- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise

6.7.2 Refresh a Work Area

The modification date is used to identify objects that have changed since the last retrieval or refresh of data.

GetFeature Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0
- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234

- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise

7 GetGMLObject Operation

This section specifies the GetGmlObject operation: its purpose, design, content, and usage, providing examples.

7.1 Purpose

The GetGmlObject operation is used to retrieve/refresh a set of features from a Data Server using object identifiers. It is intended for retrieving a specific object version or a set of versions of an object.

7.2 Design

The GetGmlObject web service request is based on the OpenGeospatial WFS specification. The implementation is based on SOAP.

7.3 Content

This section tailors the OGC Web Feature Service Implementation Specification document sections. It follows the same section ordering, and includes the applicable sections as required.

7.3.1 GetGmlObject Request Message (OGC WFS 10.0)

The Request Message contains the following XML elements:

- **Request Header:** as specified in Section 4.5.1.1.
- **Request Summary:** contains
  - **DataSourceNamespace:** identifies the Data Source uniquely.
  - **GetGmlObject Request**

7.3.2 GetGmlObject Request Introduction (OGC WFS 10.1)

The GetGmlObject operation allows retrieval of features and elements by identifier from the AVN-WS. A GetGmlObject request is processed by the AVN-WFS, and an XML document fragment, containing the result set, is returned to the client. The GetGmlObject request provides the interface through which the WFS can be asked to traverse and resolve XLinks to the features it serves.

The **<GetGmlObject>** element is used to request that a web feature service return an element with a **gml:id** attribute value specified by an **ogc:GmlObjectId**. A **GetGmlObject** element contains exactly one **<GmlObjectId>**. The value of the **gml:id** attribute on that **<GmlObjectId>** is used as a unique key to retrieve the complex element with a **gml:id** attribute with the same value.

In AIXM, features are uniquely identified by axim:identifier and a version. Need to replace <xsd:element ref="ogc:GmlObjectId"/> with an Object Reference, which would be represented as an Xlink.

The **outputFormat** attribute defines the format to use to generate the result set, and has the same options as in the GetFeature service, defined in Section 10.5.2.
The **traverseXlinkDepth, traverseXlinkExpiry** attributes have the same options as in the GetFeature service, defined in Section 10.5.2.

7.3.3 Processing (OGC WFS 10.2)

For AVN-WS, the **GetGmlObject** operation is not supported for remote resources (that is, it does not support re-direction of queries).
The data source store is searched for an element with an identifier equal to the requested ID. If no such element is found, the AVN-WS will raise an exception.

7.3.4 Response Message (OGC WFS 10.3)

The Response Message contains the following XML elements:

- **Response Header**
- **Response Summary:** see below

The format of the response to a **GetGmlObject** request is controlled by the **outputFormat** attribute.

The OGC WFS specifies that the response to a **GetGmlObject** request is the GML element returned as an XML document fragment.

For AVN-WS, the response to a **GetGmlObject** request, is the same as the response to a **GetFeature** request, a complete document containing a wfs:FeatureCollection. This provides a common response format for the clients.

7.3.5 Fault Message (OGC WFS 10.4)

The Fault Message contains the following XML elements:

- **Fault Description:** as specified in Section 4.5.3.1
- **Fault Exceptions:** as specified in Section 4.5.3.2

7.4 Usage

Scenarios where IFPA Applications use GetGmlObject are as follows:

1. Obtaining the Full description of an object when browsing.
If a Data Browser contains a Summary description of an object, GetGmlObject can be used to obtain the Full description. In this situation, for AVNIS data the ObjectVersion will be known. Quick response is desirable, so the Enterprise database should be optimized for handling a request that consists of an ObjectID and ObjectVersion.

2. Obtaining an object when assembling the object model for an instrument procedure, fix, etc.
When an existing instrument procedure is loaded into a Work Area, in order to edit it, all the other objects on which it is dependent must be loaded. To do this, the application uses GetGmlObject (Full description, providing ObjectID and ObjectVersion) to traverse the dependency graph of the object model.

3. Obtaining other versions of a particular object
If Version N of an object has been retrieved, the operator may want to view or select previous or later versions of the object. To do this, the application uses GetGmlObject, providing the ObjectID and Version.

7.5 Examples

7.5.1 Obtain the Full Description of an Object

GetGmlObject Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0

- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise

8 GetObject Operation

This section specifies the GetObject operation: its purpose, design, content, and usage, providing examples.

8.1 Purpose

The GetObject operation is used to retrieve/refresh a set of objects from a Data Server using spatial, temporal and textual filter parameters. The GetObject supports access to non-aeronautical data object types.

8.2 Design

The GetObject request is sent by the client application to the server (Data Source).

The implementation is based on SOAP.

This operation is intended to support general querying/browsing of the Data Source.

Design decisions:

1. Objects of only one ObjectType can be requested. This is done in order to simplify the processing logic on the data server side. It also prevents queries against multiple databases that would slow down the response time.

2. Spatial filter can be expressed as a location code (country and state) in order to retain IFPA database inquiry capability, as well as expressed as a bounding box.

3. Wildcarding on the ObjectName is included to match IAPA database inquiry capability.

4. Wildcarding on the ObjectName element follows SQL conventions for the LIKE operator:
    - Underscore symbol '_' means match any single character in this position.

- Percent sign '%' means match to any number of characters in this position.

8.2.1 Request Message

The Request Message contains the following XML elements:

- **Request Header**

- **Request Summary:** contains
  - **DataSourceNamespace:** identifies the Data Source uniquely.
  - **Detail:** [Full, Summary]
  - **ObjectType:** identifies a group of objects that share the identical set ofobject properties.
  - **ObjectDirectory** (required if the ObjectType is ObjectDirectory, otherwise element not allowed): identifies a particular Object Directory using a path expression, i.e., '/' separates directory levels, e.g., /Terrain/DTED5.

- **Filter** (optional, the Data Source may reject unconstrained queries for specific object types):
  - The Filter uses the same <**Filter**> element schema pattern as in the GetFeature service
  - Filter provides the following spatial options:
    - **BoundingBox:** lower left and upper right corners of geographic extent specified in latitude and longitude coordinates in WGS 84. Returns any object overlapping (interior) of bounding box.
    - **LocationCode:** country and, optionally, state.
  - Filter provides the following options, to filter object based on the values of one or more properties.
    - **Property Name:** Name of property to be filtered (e.g., ObjectName, ProductionState, EffectiveDate, ModificationDate)
    - **Literal Value**: constrained value for the property. Wildcarding symbols follow SQL conventions: '_' for single character, '%' for any number of characters.
    - **Logical operators**: AND, OR
    - **Comparison operators:** (greater than, less than or between)

8.2.2 Response Message

The Response Message contains the following XML elements:

- **Response Header:** as specified in Section 4.5.2.1.

- **Response Summary:** contains
  - **DataSourceNamespace:** identifies the Data Source uniquely.
  - **Detail:** [Full, Summary], Full may be provided even if Summary requested.
  - **ObjectType:** identifies the ObjectType of the group of objects being returned. If an ObjectDirectory contains Terrain, the ObjectType will indicate Terrain.
  - **ObjectDirectory** (required if the ObjectType is ObjectDirectory, otherwise element not allowed): identifies a particular Object Directory using a path expression, i.e., '/' separates directory levels, e.g., /Terrain/DTED.
  - **ObjectsFound:** number of objects in Object Set, may be zero.
  - **Object Set:** contains the objects matching the request.

8.2.3 Fault Message

The Fault Message contains the following XML elements:

- **Fault Description:** as specified in Section 4.5.3.1
- **Fault Exceptions:** as specified in Section 4.5.3.2

8.3 Usage

Scenarios where IFPA Application uses GetObject are as follows:

1. Retrieving the top-level Object Directories of a Data Source
The IPDS Application Data Browser requires top-level directories to organize geospatial reference data.

2. Retrieving the contents of an Object Directory
It must be possible to expand an Object Directory within the IPDS Application Data Browser, retrieving the objects within, meeting the constraints of the filter parameters.

3. Retrieving objects within a geographic area (populating a Work Area)
Requires identifying objects within a spatial bounding box.

4. Refreshing a Work Area
Requires identifying objects that have been modified since a particular date/time.

5. Retrieving objects within a geographic area matching a name
Requires wildcarding of an ObjectName.

6. Retrieving objects within a geographic area using a combination of text filter parameters.

8.4 Examples

8.4.1 Retrieve the Top-Level Object Directories of a Data Source

Retrieving from the "root" directory of the Data Source.
GetObject Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0
- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise
- Detail: Full
- ObjectType: ObjectDirectory
- ObjectDirectory: /

8.4.2 Retrieving the Contents of an Object Directory

Retrieving from the Terrain directory.

GetObject Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0
- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise
- Detail: Full
- ObjectType: ObjectDirectory
- ObjectDirectory: /Terrain

8.4.3 Retrieving Objects Within a Geographic Area

Obtaining a summary of the Aerodromes within a spatial bounding box.

GetObject Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0
- UserID: Glenn
- SessionToken: !@#$%^&*()

- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise
- Detail: Summary
- ObjectType: RadarTrack
- Filter
    - BoundingBox
    - LowerLeftCorner: N38:12:13.391 W077:24:19.626
    - UpperRightCorner: N41:32:22.877 W073:04:36.573

8.4.4 Refresh a Work Area

ModificationDate is used to identify objects that have changed since the last retrieval or refresh of data.

GetObject Request Message:

- Request Header
- InterfaceVersion: 2.1
- UserAgent: NFPG21 Windows XP 1.1 IPDS Application 2.0
- UserID: Glenn
- SessionToken: !@#$%^&*()
- RequestID: 234
- Request Summary
- DataSourceNamespace: http://us.gov.dot.faa/ato/avn/enterprise
- Detail: Summary
- ObjectType: RadarTrack
- Filter
    - BoundingBox
    - LowerLeftCorner: N38:12:13.391 W077:24:19.626
    - UpperRightCorner: N41:32:22.877 W073:04:36.573
    - Property ModificationDate greater than November 10,

9 PutFeature Operation

This section specifies the PutFeature operation: its purpose, design, content, and usage, providing examples.

9.1 Purpose

The PutFeature operation is used to store aeronautical features back to a Data Server. It submits one 'Transactable Feature Collection', a tightly coupled set of features that are to be stored in one database transaction.

9.2 Design

The PutFeature request is sent by the client application to the server (Data Source).

The implementation is based on SOAP.

Design considerations/decisions:

1. The transaction to save any individual object back into an FAA Enterprise database may fail, for reasons beyond what the requesting Application can possibly know. To simplify control and packaging of data over the interface, it is recommended that each database transaction be performed through separate invocations of a "PutFeature" Web Service, i.e., save one (or a few) features at a time.

2. A transaction order can be established from a depth-first traversal of the composition hierarchy of objects.

3. This interface control document specifies what the transaction units must be, primarily based on an analysis of FAA Enterprise needs. There are situations where a set (directed acyclic graph) of objects may need to be submitted. This set of features is called a 'Transactable Feature Collection'.

4. The Operator determines whether a submission is an update or amendment. Each Transactable Feature Collection submitted is tagged accordingly. Objects submitted back to the AVN Enterprise fall into one of the following categories:

    a) An update (save/overwrite existing version/amendment), if object already exists in the Enterprise. User must be authorized to update it.

    b) An amendment if object already exists in the Enterprise.

    c) A completely new object, original version.

5. An original (completely new) object does not have a Data Source ObjectID when submitted. The recipient Data Source needs to manufacture its own ObjectID (and ObjectVersion) and provide this information back to the requesting Application. There may also be other object properties that need to be manufactured by the Data Source and returned, e.g., EffectiveDate. For a Transactable Feature Collection, the Data Source needs to manufacture and provide this information for each new object in the set.

6. The FAA Enterprise is responsible for assigning the Effective Date for publication. It sets this property on all original and amended objects. The assigned EffectiveDate must be returned to the calling Application.

7. Future states: 'for publishing', future (multiple allowed), or ultimate (only one). FAA Enterprise derives this state value on submission of data from the associated PTS Workflow

9.2.1 Transactable Feature Collections

The Table 13-1 lists the allowed feature type collections, and the types of feature in each collection.

Table 9.2.1 PutFeature Collections

Feature Collection Type	**AIXM Feature Types Allowed in the Feature Collection**	**Comments**
AerodromeHeliport		
	AerodromeHeliport	
Runway		
	Runway	
	RunwayDirection	
	RunwayCenterlinePoint	Feature Collection must include a RunwayDirection feature
	RunwayDeclaredDistance	Feature Collection must include a RunwayDirection feature
	RunwayProtectArea	Feature Collection must include a RunwayDirection feature
	RunwayVisualRange	Feature Collection must include a RunwayDirection feature
	VisualGlideSlopeIndicator	
	RunwayDirectionLightSystem	
	ApproachLightingSystem	
ServiceAtAerodromeHeliport		
	ServiceAtAerodromeHeliport	
	Frequency	Feature Collection must include a ServiceAtAerodromeHeliport feature
Navaid		
	Navaid	

	NavaidEquipment	
	RadioFrequencyLimitation	
Designated Point		
	DesignatedPoint	**Feature Collection Type**
	PointReference (Module 2)	**DesignatedPoint**
	InstrumentApproachProcedure	
	ServiceOnInstrumentApproachProce dure	
	CirclingArea	
	TerminalArrivalArea	
	ProcedureTransition	
	SegmentLeg	
	ArrivalFeederLeg	
	InitialLeg	
	IntermediateLeg	
	FinalLeg	
	MissedApproachLeg	
	HoldingPattern	
	SafeAltitudeArea	

9.2.2 Request Message

The Request Message contains the following XML elements:

- **Request Header:** as specified in Section 4.5.1.1.
- **Request Summary:** contains
  - **DataSourceNamespace:** identifies the Data Source uniquely.
  - **ProjectID:** unique identifier for the PTS Project.
  - **ProjectName:** name of the Project, to help later diagnosis or auditing of the request.
  - **Feature Type:** identifies the type of the Transactable Feature Collection
  - **Feature Count:** the number of feature instances in the Transactable FeatureCollection, must be greater than zero.
- **Feature Collection:** for each feature timeslice in the Feature Collection, the following is provided:
- **Operation:** [Create, Update, Amend, Delete], with the following semantics:

  - *Create*: new/original object being submitted. Its ObjectID and ObjectVersion properties provided have been manufactured outside of the recipient Data Source.

• *Update*: changes to an object version that already exists in the Data Source. The ObjectID and ObjectVersion provided are those of the recipient Data Source. The ObjectVersion is not changed as a result of an Update operation.

• *Amend*: a revision of an existing object, that necessitates a new object version. The ObjectID and ObjectVersion provided are those of the object taken from the Data Source, i.e., the ObjectVersion has not been changed. The recipient Data Source manufactures a new ObjectVersion.

• *Delete:* notice to remove an object from the Data Source. No top-level object can be deleted; however a subservient child object can be deleted if and only if the Operation on its parent is *Update.*

• **Feature Properties:** full description of object including all essential Object Reference properties.

9.2.3 Response Message

The Response Message contains the following XML elements:

• **Response Header:** as specified in Section 4.5.2.1.
• **Response Summary:** states for each feature instance in the Transactable Feature Collection:
• **Feature Receipt**
• **Operation:** [Create, Update, Amend, Delete], as submitted.
• **Feature Type:** identifies the type of the feature, as submitted.
• **Client Object Reference**
• **Namespace:** as submitted.
• **ObjectID**: as submitted.
• **ObjectVersion**: as submitted.
• **ObjectName**: as submitted.
• **Data Source Object Reference:** includes properties that may be manufactured by the Data Source
• **Namespace:** as manufactured by Data Source: a new value from what was submitted for Create operation.
• **ObjectID**: as manufactured by Data Source: a new value from what was submitted for Create operation.
• **ObjectVersion**: as manufactured by Data Source: a new value from what was submitted for Create and Amend operations.
• **ObjectName**: as submitted.

• **ProductionState** (optional): any combination of the set of [Working, Pending, Active, History, Future].
• **EffectiveDate** (optional): expressed as a GML TimeInstant.
• **ModificationDate** (optional): expressed as a GML TimeInstant.

9.2.4 Fault Message

The Fault Message contains the following XML elements:
• **Fault Description:** as specified in Section 4.5.3.1
• **Fault Exceptions:** as specified in Section 4.5.3.2

9.3 Usage

The typical fashion in which PutFeature is intended to be used for the purpose of storing data back into the FAA AVNIS data server is suggested below:

1. The operator uses the client application to create and edit data objects.

2. The operator uses the client application to prepare the data objects to submit to a Data Source. This includes specifying:
   a) The Project under which the changes are being submitted.
   b) For every changed feature, whether it is an amendment or an update.

3. Once the operator confirms that the data is to be submitted, the client application determines the first Transactable Feature Collection to be sent to the Data Source, based on the established order of precedence
   a) For new objects, the client application provides manufactured ObjectID and ObjectVersion.

4. The client application issues the PutFeature request containing the first Transactable Feature Collection.

5. The FAA Enterprise data server validates the PutFeature request.
   a) The Project ID is validated, and the submission only proceeds if the Project has been assigned to the UserID.
   b) The Transactable Feature Collection is validated:
      1. For an *update* operation, the data server verifies that the user has update authorization.
      2. For some ObjectTypes, the data server checks object-specific properties, e.g., the name of a Fix within CONUS, to ensure it is on the available Fix Name List.

6. The FAA Enterprise data server manufactures unspecified object properties. For example,

a) ProductionState: based on the characteristics of the Project, the values for object status are set, e.g., working, future, etc.
b) EffectiveDate: based on the target publication date for the Project, the object's effective date is set.

7. The FAA Enterprise data server inserts, updates, deletes the objects in the Transactable Feature Collection as a single transaction. As part of the transaction, the ObjectID and ObjectVersion are handled as follows:
a) *Create* operation: the data server manufactures a new unique ObjectID and the ObjectVersion is set as 'original'.
b) *Update* operation: the data server uses the ObjectID and ObjectVersion provided.
c) *Amend* operation: the data server re-uses the ObjectID and manufactures the next ObjectVersion.
d) *Delete* operation: the data server uses the ObjectID and ObjectVersion provided.

8. The FAA Enterprise data server completes any follow-on "book-keeping" resulting from the transaction. For example,
a) The Project is updated, adding Object References, possibly adding Tasks.
b) Object-specific book-keeping is performed, e.g., removing the name used for a new Fix from the Available Fix Names List.

9. The FAA Enterprise data server assembles the PutFeature response, including object properties it has manufactured, e.g., ModificationDate.

10. The client application unpacks the PutFeature response and updates its copies of the objects submitted with the properties manufactured by the Data Source: one or more of ObjectID, ObjectVersion, ProductionState, EffectiveDate, ModificationDate, etc.

11. The client application proceeds to the next Transactable Feature Collection in the order of precedence, returning to step 4, until all have been stored on the FAA Enterprise data server.

10 PutObject Operation

This section specifies the PutObject operation: its purpose, design, content, and usage, providing examples.

10.1 Purpose

The PutObject operation is used to store non-aeronautical objects back to a Data Server.

It submits one 'Transactable Object Set', a tightly coupled set of objects that are to be stored in one database transaction.

10.2 Design

The PutObject request is sent by the client application to the server (Data Source).

The implementation is based on SOAP.

1. The PutObject service has similar design considerations/decisions to the PutFeature service.

2. A transaction order can be established from a depth-first traversal of the composition hierarchy of objects managed by the IFPA Applications.

3. This interface control document specifies what the transaction units must be, this set of objects is called a 'Transactable Object Set'.

4. The Operator determines whether a submission is a new object, update or amendment. Each Transactable Object Set submitted is tagged accordingly.

5. Every submission back to the FAA Enterprise must be associated with a Workflow Project.

6. The FAA Enterprise is responsible for validating the Workflow Project, and performing any necessary updates within PTS to link Workflow objects to the new or amended data objects in the Enterprise.

7. Future states: 'for publishing', future (multiple allowed), or ultimate (only one). FAA Enterprise derives this state value on submission of data from the associated PTS Workflow.

8. The workflow for aeronautical objects are by association with the Fix / SIAP, etc that they are linked to.

10.2.1 Request Message

The Request Message contains the following XML elements:

- **Request Header:** as specified in Section 4.5.1.1.
- **Request Summary:** contains
- **DataSourceNamespace:** identifies the Data Source uniquely.
- **ProjectID:** unique identifier for the Project.
- **ProjectName:** name of the Project, to help later diagnosis or auditing of the request.

• **ObjectType:** identifies the type of the Transactable Object Set.
• **ObjectCount:** the number of objects in the Transactable Object Set, must be greater than zero.
• **Object Set:** for each object, the following is provided:
• **Operation:** [Create, Update, Amend, Delete], with the following semantics:

• *Create*: new/original object being submitted. Its ObjectID and ObjectVersion properties provided have been manufactured outside of the recipient Data Source.

• *Update*: changes for an object version that already exists in the Data Source. The ObjectID and ObjectVersion provided are those of the recipient Data Source.

• *Amend*: changes that necessitate a new object version. The ObjectID and ObjectVersion provided are those of the object taken from the Data Source, i.e., the ObjectVersion has not been changed, and the recipient Data Source must manufacture a new ObjectVersion.

• *Delete:* notice to remove an object from the Data Source. Depending on the object type, the FAA Enterprise may explicitly delete the object, or treat a delete as a cancellation.

• **Object Properties:** full description of object including all essential Object Reference properties.

Note: For an object being deleted, only need its Object Reference, not a full description.

10.2.2 Response Message

The Response Message contains the following XML elements:

• **Response Header:** as specified in Section 4.5.2.1.
• **Response Summary:** states for each object in the Transactable Object Set:
• **Object Receipt**
• **Operation:** [Create, Update, Amend, Delete], as submitted.
• **ObjectType:** identifies the type of the Object, as submitted.
• **Client Object Reference**
• **Namespace:** as submitted.
• **ObjectID**: as submitted.
• **ObjectVersion**: as submitted.
• **ObjectName**: as submitted.
• **Data Source Object Reference:** includes properties that may be manufactured by the Data Source
• **Namespace:** as manufactured by Data Source: a new value

from what was submitted for Create operation.
• **ObjectID**: as manufactured by Data Source: a new value from what was submitted for Create operation.
• **ObjectVersion**: as manufactured by Data Source: a new value from what was submitted for Create and Amend operations.
• **ObjectName**: as submitted.
• **ProductionState** (optional)**:** any combination of the set of [Working, Pending, Active, History, Future].
• **EffectiveDate** (optional)**:** may be expressed as a GML TimeInstant.
• **EndDate** (optional)**:** may be expressed as a GML TimeInstant.
• **ModificationDate** (optional)**:** may be expressed as a GML TimeInstant.

10.2.3 Fault Message

The Fault Message contains the following XML elements:

• **Fault Description:** as specified in Section 4.5.3.1
• **Fault Exceptions:** as specified in Section 4.5.3.2

10.3 Usage

The typical fashion in which PutObject is intended to be used for the purpose of storing data back into the FAA AVNIS data server is suggested below:

1. The operator uses the client application to create and edit data objects.

2. The operator uses the client application to prepare the data objects to submit to a Data Source.
This includes specifying:
a) The Project under which the changes are being submitted.
b) For every changed Transactable Object, whether it is an amendment or an update.

3. Once the operator confirms that the data is to be submitted, the client application determines the first Transactable Object Set to be sent to the Data Source, based on the established order of precedence
a) For new objects, the client application provides manufactured ObjectID and ObjectVersion.

4. The client application issues the PutObject request containing the first Transactable Object Set.

5. The FAA Enterprise data server validates the PutObject request.
   a) The Project ID is validated, and the submission only proceeds if the Project has been assigned to the UserID.
   b) The Transactable Object Set is validated:
      1. For an *update* operation, the data server verifies that the user has update authorization.
      2. For some ObjectTypes, the data server checks object-specific properties, e.g., the Waiver justification is complete

6. The FAA Enterprise data server manufactures unspecified object properties. For example,
   a) ProductionState: based on the characteristics of the Project, the values for object status are set, e.g., working, future, etc.
   b) EffectiveDate: based on the target publication date for the Project, the object's effective date is set.

7. The FAA Enterprise data server inserts, updates, deletes the objects in the Transactable Object Set as a single transaction. As part of the transaction, the ObjectID and ObjectVersion are handled as follows:
   a) *Create* operation: the data server manufactures a new unique ObjectID and the ObjectVersion is set as 'original'.
   b) *Update* operation: the data server uses the ObjectID and ObjectVersion provided.
   c) *Amend* operation: the data server re-uses the ObjectID and manufactures the next ObjectVersion.
   d) *Delete* operation: the data server uses the ObjectID and ObjectVersion provided.

8. The FAA Enterprise data server completes any follow-on "book-keeping" resulting from the transaction. For example,
   a) The Project is updated, adding Object References, possibly adding Tasks.
   b) Object-specific book-keeping is performed.

9. The FAA Enterprise data server assembles the PutObject response, including object properties it has manufactured, e.g., ModificationDate.

10. The client application unpacks the PutObject response and updates its copies of he objects submitted with the properties manufactured by the Data Source: one or ore of ObjectID, ObjectVersion, ProductionState, EffectiveDate, ModificationDate, etc.

11. The client application proceeds to the next Transactable Object Set in the order of precedence, returning to step 4, until all have been stored on the FAA Enterprise data server.

11 GetConfigData Operation

This section specifies the GetConfigData operation: its purpose, design, content, and usage, providing examples.

11.1 Purpose

The GetConfigData operation is used to retrieve new or updated configuration data from the Data Server.

11.2 Context

This service supports the following:

- Country Code Reference Table
- Aerodrome Name Reference Table

11.3 Design

The GetConfigData request is sent by the client application to the server (Data Source).

The implementation is based on SOAP.

11.3.1 Request Message

The Request Message contains the following XML elements:

- **Request Header:** as specified in Section 4.5.1.1.
- **Request Summary:** contains
    - **ConfigurationSetName:** either the preset term 'Master Index' or any name assigned to a Configuration Set.
    - **ModificationDate:**

11.3.2 Response Message

The Response Message contains the following XML elements:

- **Response Header**
- **Response Summary:** contains
    - **ConfigurationSetName:** either the preset term 'Master Index' or any name assigned to a Configuration Set.
    - **ModificationDate:** date Configuration Set was last modified.
    - **Attachments:** contains Configuration Set.

11.3.3 Fault Message

The Fault Message contains the following XML elements:

- **Fault Description:** as specified in Section 4.5.3.1
- **Fault Exceptions:** as specified in Section 4.5.3.2

12 AIXM-5 Schema Usage

This section specifies general rules for encoding AIXM features for the AVN WS. This enables both sides of interface to encode data in the same consistent manner.

12.1 Natural Keys in AIXM-5 Xlinks

Wherever a feature has an xlink (object reference) to one of the feature types, the XPath also includes a natural key that identifies the referenced feature.

Table 12-1 XPath Natural Key

AIXM Feature	Natural Key Attributes
AerodromeHeliport	designator
Runway	designator AerodromeHeliport designator
RunwayDirection	designator AerodromeHeliport designator
ServiceAtAerodromeHeliport	type AerodromeHeliport - designator
Navaid	type identifier countryCode
NavaidEquipment	type identifier countryCode
DesignatedPoint	designator countryCode

SafeAltitudeArea	safeAreaType protects_InstrumentApproachProcedure (xlink to InstrumentApproachProcedure) basedOn_DesignatedPoint (xlink to DesignatedPoint) basedOn_Navaid (xlink to Navaid)
HoldingPattern	type holdFixType outboundCourse outboundCourseType turnDirection Navaid - type, identifier, countryCode DesignatedPoint - designator, countryCode
Obstacle	name type
InstrumentApproachProcedure	designCriteria name AerodromeHeliport - designator RunwayDirection - designator

12.2 AIXM Types

This section specifies general rules for elements in AIXM features.

12.2.1 Empty Tags and Nil Attributes

In GetFeature and GetGmlObject response messages, empty tags are not sent.
In PutFeature request messages, empty tags are not sent. Elements are handled by FAA Enterprise as follows:

Insert:
Missing elements in the message are set to undefined in the FAA Enterprise. An undefined property may also have its element attribute set to undefined (xsi:nil), and are also set to undefined by the FAA Enterprise.

Update:
Elements must be set for all updated fields. The FAA Enterprise only updates the object properties that are included in the message, missing elements in the message will leave the existing value unchanged in the FAA Enterprise. A client may send elements for unchanged fields, the FAA Enterprise also updates these object properties. If a property is changed to undefined, its element attribute must be set to undefined (xsi:nil), and it is set to undefined by the FAA Enterprise.

Amend:
An amend operation creates a new version of a feature. The process of creating the new version is a copy followed by update. The FAA Enterprise makes a copy of the object being updated, and then applies the update based on the message contents. Hence the client need only send the amended properties (it may send elements for unchanged fields).

Empty tags should not be sent. Elements must be set for all updated fields. The FAA Enterprise only updates the object properties that are included in the message, missing elements in the message leave the existing values unchanged in the FAA Enterprise. If a property is changed to undefined, its element attribute must be set to undefined (xsi:nil), and it is set to undefined by the FAA Enterprise.

12.2.2 aixm:_Feature Elements

The aixm:_Feature elements are used as follows:

- element gml:description: may be filled in by a client for debug purposes, but is not used by the FAA Enterprise. The AVN-WS will not populate this element.

- element gml:name: may be filled in by a client for debug purposes, but is not used by the FAA Enterprise. The AVN-WS will not populate this element.

- element gml:boundedBy: is not used.

- group aixm:StandardAIXMFeatureProperties: is used. It has the aixm:identifier.

- group aixm:DynamicFeatureProperties: is not used (it is a gml:validTime for a feature)

- element featureMetadata (type="aixm:FeatureMetadataPropertyType"): feature level metadata, see below.

12.2.3 aixm:AIXMTimeSliceType Elements

The aixm:AIXMTimeSliceType elements (see Figure 17-2) are used as follows:

- element gml:validTime: time element are coded as a gml:TimePeriod for example:

```
<gml:validTime>
<gml:TimePeriod>
<gml:beginPosition>2004-01-15T00:00:00</gml:beginPosition>
<gml:endPosition indeterminatePosition="unknown" />
</gml:TimePeriod>
</gml:validTime>
```

- The gml:TimePosition frame and calendarEraName attributes are not used, the Gregorian calendar with UTC is the default reference system. The gml:TimePosition element is the dateTime built-in simple type.

• The gml:TimePosition indeterminatePosition attribute is set to "unknown" when no specific value for temporal position is provided.

• element aixm:interpretation: always either BASELINE, or PERM DELTA (when operation is amend)

• element aixm:sequenceNumber: is not used

• element aixm:correctionNumber: is not used.

• element timeSliceMetadata (type="aixm:FeatureTimeSliceMetadataPropertyType)

12.3 MetaData

Table 12-3 lists the metadata elements that are required. It includes the name of the mandatory element and its description, as well as how it maps to the ISO19115 standard. For core geographic datasets per ISO19115, (M) indicates that the element is mandatory and (C) indicates that the element is conditionally mandatory. For reference, in the third column of the table, we include the short name of the element and the row number corresponding to Annex B.2 (Metadata package data dictionaries) of ISO19115. For example, *dateStamp* from the MessageMetadata class is the date on which the metadata for the AIXM message was compiled. The *dateStamp* element is mandatory for geographic datasets per ISO19115, referred to as mdDateSt, and can be found in row 9 of Annex B.2 in ISO19115. For those elements that are not in ISO19115, the justification for a data element extension is stated.

Table 12-3 Mandatory AIXM Metadata Elements

Mandatory AIXM Metadata Element	**Description**	**Mapping to ISO19115**
FeatureTimeSliceMetadata dateStamp	Date on which the metadata for the feature timeslice was compiled.	ISO 19115 (M) mdDateSt (9)
MessageMetadata > Contact >systemName FeatureTimeSliceMetadata > Contact >systemName	Name of the responsible system (i.e., database, or repository that transmitted or compiled info). If organization or individual name not available, must include system name.	ISO19115 data element extension - Needed a new element within the responsible party class to describe the responsible system.
MessageMetadata > Contact >organizationName FeatureTimeSliceMetadata > Contact >organizationName	Name of the responsible organization. If individual or system name not available, must include organization name.	ISO19115 (C) rpOrgName (376)

MessageMetadata >messageIdentificationInfo > abstract	Brief narrative summary on the contents of the AIXMmessage. Contents can include multiple features and operating instructions on how to use the feature data.	ISO19115 (M) idAbs(25)
MessageMetadata >messageIdentificationInfo > language	The language used within the AIXM message. Follows ISO639-2. Best practice recommends the language to be English.	ISO19115 (M) dataLang (39). In ISO19115, language has multiplicity. We restrict domain to 1 language in AIXM model.

12.4 GML Types

12.4.1 Coordinate Reference System

A coordinate system is as set of (mathematical) rules for specifying how coordinates are to be assigned to points. A coordinate reference system is a coordinate system that is related to the real world by a datum. GML (and AIXM-5) requires a coordinate reference system (CRS) to be referenced whenever location coordinate information is given. CRS provides the meaning for location coordinates. The CRS is generally given using the srsName (Spatial Reference System name) attribute of the GML elements, this allows the CRS specified at each instance to be different.

Propose (from OGC document "URNs of definitions in ogc namespace", OGC 05-010) that the URN value for an any URI that references an object in the European Petroleum Survey Group (EPSG) database has the form: urn:ogc:def:objectType:EPSG:version:code The "authority" part of a URN is "EPSG". The "code" part of a URN should be the EPSG "code" unique identifier of the referenced definition. Alternately, the "code" part of a URN can be the EPSG "name" unique identifier. The "version" part will be included in this case, since the EPSG sometimes deprecates and replaces existing definitions (latest version of EPSG is 6.12).

For example, below specifies the co-ordinates are in WGS84:

```
<gml:Envelope srsName=" urn:ogc:def:crs:EPSG:6.12:4326">
<gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
<gml:upperCorner>35.5472 42.8143</gml:upperCorner>
</gml:Envelope>
```

The datums and their EPSG codes that are used by the AVN-WS are listed in Table 12.4.

Table 12-4 SRS EPSG Code

Horizontal Datum Item	DVOF Horizontal Datum Code	EPSG Code
Adindan, Sudan	ADI	4201
Arc 1950, Africa	ARF	4209
Arc 1960	ARS	4210
Ascension Island 1958	ASC	4712
Corrego Alegre, Brazil	COA	4225
European	EUR	4258
Guadeloupe Astro 1946	GUD	????
Heart North, Afghanistan	HEN	4255
Hjorsey 1955, Iceland	HJO	4658
Indian	IND	4239 (Indian 1954) 4240 (Indian 1975)
Ireland 1965	IRL	4299
Keratu	KEA	4245
Local (Local Astro)	LOC	????
Luzon, Philippines	LUZ	4253
Merchich, Morocco	MER	4261
Nahwran	NAH	4270
Nigerial	NIG	4263
North American 1927	NAS	4267
Ordnance Survey of Great Britain 1936	OGB	4277
Provisional South American 1956	PRP	4248
Timbalai 1948, Borneo	TIL	4298
Tokyo	TOK	4301
Undetermined	UND	????
Wake-Eniwetok 1960	ENW	4732
Viti-Levu	VTL	4731
World Geodetic System 1966	WGB	????
World Geodetic System 1972	WGC	4322
World Geodetic System 1984	WGD	4326
World Geodetic System 1984	WGE	4326
Datum not listed in DVOF Manual	ZZZ	

12.4.2 GM_Point (gml:PointType)

The gml:StandardObjectProperties group is not used (it contains elements gml:metaDataProperty, gml:description, gml_name).

The element gml:pos is used for all points (the other choice options gml:coord, and gml:coordinates are deprecated with GML Version 3).

The gml:pos attributes is be used as follows:

- srsName: is as specified above
- srsDimension: is not used (is always 2)
- axisLabels: is not used
- uomLabels: is not used

Coordinates are always geodetic latitude, followed by geodetic longitude, in decimal degrees, North positive, East positive.

12.5 Notes Object

12.5.1 Annotation Types

The AIXM Note object basically allows for only a textual annotation to a feature or object. For AVN-WS, the AIXM Note object is extended to provide different types of annotation:

Comment: A comment annotation has following attributes:

- Topic: The topic of the comment (see Table 17-4below).
- Text: The comment text (limited to 2000 characters).
- Date: Date added.
- Priority: The priority or sort order for printing. Also referred to as Print Sequence.
- Feature_type (on ENTITY_COMMENT from COMMUNICATION)??

User Entered Standard Note: A user entered standard note has following attributes:

- Topic: Equipment, Final Approach Course, Frequency, Holding, Obstacle, Procedure, Profile, Profile2, Altimeter, Briefing, Briefing2, Plan/Prof, Planview.
- Print Block: Additional Flight Data, Final Segment, Notes A, Notes B.
- Chart Note: Boolean (codeYesNo).
- Text: Text of note.

Standard Note: A standard note has following attributes:

- Note Control Number (from reference table).
- Parameters: Parameters to substitute in standard note text.
- Text: The completed note text.

Attachment:

- Topic: Describes attachment (limited to 2000 characters)
- File Size: Size of attachment (Kbyte)
- File Name: Attachment file name
- URL: Attachment file location
- Creator: Name of attachment creator

The Note object propertyName element is not used.